Mellanox OFED for Linux User Manual

Rev 1.5.3-3.1.0

Last Modified June 26, 2013

www.mellanox.com

NOTE:
THIS HARDWARE, SOFTWARE OR TEST SUITE PRODUCT (“PRODUCT(S)”) AND ITS RELATED DOCUMENTATION ARE PROVIDED BY MELLANOX TECHNOLOGIES “AS-IS” WITH ALL FAULTS OF ANY KIND AND SOLELY FOR THE PURPOSE OF AIDING THE CUSTOMER IN TESTING APPLICATIONS THAT USE THE PRODUCTS IN DESIGNATED SOLUTIONS. THE CUSTOMER'S MANUFACTURING TEST ENVIRONMENT HAS NOT MET THE STANDARDS SET BY MELLANOX TECHNOLOGIES TO FULLY QUALIFY THE PRODUCTO(S) AND/OR THE SYSTEM USING IT. THEREFORE, MELLANOX TECHNOLOGIES CANNOT AND DOES NOT GUARANTEE OR WARRANT THAT THE PRODUCTS WILL OPERATE WITH THE HIGHEST QUALITY. ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT ARE DISCLAIMED. IN NO EVENT SHALL MELLANOX BE LIABLE TO CUSTOMER OR ANY THIRD PARTIES FOR ANY DIRECT, INDIRECT, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES OF ANY KIND (INCLUDING, BUT NOT LIMITED TO, PAYMENT FOR PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY FROM THE USE OF THE PRODUCT(S) AND RELATED DOCUMENTATION EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Mellanox Technologies
350 Oakmead Parkway Suite 100
Sunnyvale, CA 94085
U.S.A.
www.mellanox.com
Tel: (408) 970-3400
Fax: (408) 970-3403

Mellanox Technologies, Ltd.
Beit Mellanox
PO Box 586 Yokneam 20692
Israel
www.mellanox.com
Tel: +972 (0)74 723 7200
Fax: +972 (0)4 959 3245

© Copyright 2013. Mellanox Technologies. All Rights Reserved.

Mellanox®, Mellanox logo, BridgeX®, ConnectX®, CORE-Direct®, InfiniBridge®, InfiniHost®, InfiniScale®, MLNX-OS®, PhyX®, SwitchX®, UFM®, Virtual Protocol Interconnect® and Voltaire® are registered trademarks of Mellanox Technologies, Ltd.

Connect-IB™, FabricIT™, Mellanox Open Ethernet™, Mellanox Virtual Modular Switch™, MetroX™, MetroDX™, ScalableHPC™, Unbreakable-Link™ are trademarks of Mellanox Technologies, Ltd.

All other trademarks are property of their respective owners.

Table of Contents

Table of Contents 3
List of Tables 8
Chapter 1 Mellanox OFED Overview 14
1.1 Introduction to Mellanox OFED 14
1.2 Introduction to Mellanox VPI Adapters 14
1.3 Mellanox OFED Package 14
1.3.1 ISO Image 14
1.3.2 Software Components 15
1.3.3 Firmware 16
1.3.4 Directory Structure 16
1.4 Architecture 17
1.4.1 mthca HCA (IB) Driver 17
1.4.2 mlx4 VPI Driver 18
1.4.3 Mid-layer Core 18
1.4.4 ULPs 18
1.4.5 MPI 19
1.4.6 InfiniBand Subnet Manager 19
1.4.7 Diagnostic Utilities 20
1.4.8 Mellanox Firmware Tools 20
1.5 Quality of Service 20
Chapter 2 Installation 21
2.1 Hardware and Software Requirements 21
2.1.1 Hardware Requirements 21
2.1.2 Software Requirements 22
2.2 Downloading Mellanox OFED 22
2.3 Installing Mellanox OFED 22
2.3.1 Pre-installation Notes 23
2.3.2 Installation Script 23
2.3.3 Installation Procedure 26
2.3.4 Installation Results 31
2.3.5 Post-installation Notes 33
2.4 Updating Firmware After Installation 33
2.5 Uninstalling Mellanox OFED 34
Chapter 3 Configuration Files 35
Chapter 4 Driver Features 36
4.1 RDMA over Converged Ethernet 36
4.1.1 RoCE Overview 36
4.1.2 Software Dependencies 36
4.1.3 Firmware Dependencies 36
4.1.4 General Guidelines 36
4.1.5 Ported Applications 37

4.1.6 GID Tables 37
4.1.7 Using VLANs 37
4.1.8 Reading Port Counters Statistics 38
4.1.9 A Detailed Example 39
4.1.10 Configuring DAPL over RoCE 44
4.2 Reliable Datagram Sockets 45
4.2.1 Overview 45
4.2.2 RDS Configuration 45
4.3 Sockets Direct Protocol 45
4.3.1 Overview 45
4.3.2 libsdp.so Library 46
4.3.3 Configuring SDP 46
4.3.4 Environment Variables 49
4.3.5 Converting Socket-based Applications 49
4.3.6 BZCopy – Zero Copy Send 55
4.3.7 Using RDMA for Small Buffers 55
4.4 SCSI RDMA Protocol 56
4.4.1 Overview 56
4.4.2 SRP Initiator 56
4.5 Ethernet over IB (EoIB) vNic 63
4.5.1 Ethernet over IB Topology 64
4.5.2 EoIB Configuration 65
4.5.3 Retrieving EoIB Information 69
4.5.4 Advanced EoIB Settings 74
4.6 IP over InfiniBand 78
4.6.1 Introduction 78
4.6.2 IPoIB Mode Setting 79
4.6.3 IPoIB Configuration 79
4.6.4 Subinterfaces 83
4.6.5 Verifying IPoIB Functionality 84
4.6.6 Bonding IPoIB 84
4.7 Quality of Service 85
4.7.1 Quality of Service Overview 85
4.7.2 QoS Architecture 86
4.7.3 Supported Policy 87
4.7.4 CMA Features 88
4.7.5 OpenSM Features 89
4.8 Atomic Operations 89
4.8.1 Enhanced Atomic Operations 89
4.9 Huge Pages Support for Queue Resources 90
Chapter 5 VPI Configuration and Management 92
5.1 Port Type Management 92
5.2 Auto Sensing 93
5.2.1 Enabling Auto Sensing 93
5.3 InfiniBand Driver 93

5.4 Ethernet Driver 93
5.4.1 Overview 93
5.4.2 Loading the Ethernet Driver 94
5.4.3 Unloading the Driver 94
5.4.4 Ethernet Driver Usage and Configuration 94
Chapter 6 Performance 97
6.1 General System Configurations 97
6.1.1 PCI Express (PCIe) Capabilities 97
6.1.2 BIOS Power Management Settings 97
6.1.3 Intel® Hyper-Threading Technology 97
6.2 Improving IPv4/IPv6 Traffic Performance 98
6.2.1 Tuning the Network Adapter for Improved IPv4 Traffic Performance 98
6.2.2 Tuning the Network Adapter for Improved IPv6 Traffic Performance 98
6.2.3 Interrupt Moderation 98
6.2.4 IRQ Affinity 99
6.2.5 Preserving Your Performance Settings After A Reboot 100
6.3 Performance Troubleshooting 101
6.3.1 PCI Express Performance Troubleshooting 101
6.3.2 InfiniBand Performance Troubleshooting 102
6.3.3 System Performance Troubleshooting 103
Chapter 7 MPI - Message Passing Interface 104
7.1 Overview 104
7.2 Prerequisites for Running MPI 104
7.2.1 SSH Configuration 104
7.3 MPI Selector - Which MPI Runs 105
7.4 Compiling MPI Applications 106
Chapter 8 MellanoX Messaging 107
8.1 Enabling MXM in OpenMPI 107
Chapter 9 OpenSM – Subnet Manager 108
9.1 Overview 108
9.2 opensm Description 108
9.2.1 opensm Syntax 108
9.2.2 Environment Variables 115
9.2.3 Signaling 115
9.2.4 Running opensm 115
9.3 osmtest Description 116
9.3.1 Syntax 116
9.3.2 Running osmtest 118
9.4 Partitions 119
9.4.1 File Format 119
9.5 Routing Algorithms 121
9.5.1 Effect of Topology Changes 122
9.5.2 Min Hop Algorithm 123
9.5.3 UPDN Algorithm 123

9.5.4 Fat-tree Routing Algorithm 124
9.5.5 LASH Routing Algorithm. 126
9.5.6 DOR Routing Algorithm 127
9.5.7 Torus-2QoS Routing Algorithm 128
9.6 Quality of Service Management in OpenSM 136
9.6.1 Overview. 136
9.6.2 Advanced QoS Policy File 137
9.6.3 Simple QoS Policy Definition 138
9.6.4 Policy File Syntax Guidelines 138
9.6.5 Examples of Advanced Policy File 138
9.6.6 Simple QoS Policy - Details and Examples 141
9.6.7 SL2VL Mapping and VL Arbitration 143
9.6.8 Deployment Example 145
9.7 QoS Configuration Examples 145
9.7.1 Typical HPC Example: MPI and Lustre 146
9.7.2 EDC SOA (2-tier): IPoIB and SRP 146
9.7.3 EDC (3-tier): IPoIB, RDS, SRP. 147
9.8 Adaptive Routing 149
9.8.1 Overview. 149
9.8.2 Installing the Adaptive Routing 149
9.8.3 Running Subnet Manager with Adaptive Routing Manager. 149
9.8.4 Querying Adaptive Routing Tables 150
9.8.5 Adaptive Routing Manager Options File. 150
9.9 Congestion Control 153
9.9.1 Congestion Control Overview. 153
9.9.2 Running OpenSM with Congestion Control Manager. 154
9.9.3 Configuring Congestion Control Manager 154
9.9.4 Configuring Congestion Control Manager Main Settings 155
Chapter 10 InfiniBand Fabric Diagnostic Utilities 158
10.1 Overview 158
10.2 Utilities Usage 158
10.2.1 Common Configuration, Interface and Addressing 158
10.2.2 IB Interface Definition. 159
10.2.3 Addressing 159
10.3 ibdiagnet (of ibutils2) - IB Net Diagnostic 160
10.3.1 SYNOPSYS 160
10.3.2 Output Files 161
10.3.3 Return Codes. 162
10.4 ibdiagnet (of ibutils) - IB Net Diagnostic 162
10.4.1 SYNOPSYS 162
10.4.2 Output Files 164
10.4.3 ERROR CODES 165
10.5 ibdiagpath - IB diagnostic path 165
10.5.1 SYNOPSYS 165
10.5.2 Output Files 167

10.5.3 ERROR CODES 167
10.6 ibv_devices 167
10.7 ibv_devinfo 168
10.8 ibdev2netdev 169
10.8.1 SYNOPSYS 169
10.9 ibstatus 170
10.10 ibportstate 172
10.11 ibroute 175
10.12 smpquery 178
10.13 perfquery 182
10.14 ibcheckerrs 184
10.15 mstflint 186
10.16 ibv_asyncwatch 190
10.17 ibdump 190

Appendix A Mellanox FlexBoot 193
A.1 Overview 193
A.2 Burning the Expansion ROM Image 194
A.3 Preparing the DHCP Server in Linux Environment 195
A.4 Subnet Manager – OpenSM 197
A.5 TFTP Server 197
A.6 BIOS Configuration 197
A.7 Operation 197
A.8 Command Line Interface (CLI) 199
A.9 Diskless Machines 200
A.10 iSCSI Boot 206
A.11 WinPE 207

Appendix B SRP Target Driver 208
B.1 Prerequisites and Installation 208
B.2 How-to run 208
B.3 How-to Unload/Shutdown 211

Appendix C mlx4 Module Parameters 212
C.1 mlx4_core Parameters 212
C.2 mlx4_ib Parameters 212
C.3 mlx4_en Parameters 213

Appendix D ib-bonding Driver for Systems Using SLES10 SP4 214
D.1 Using the ib-bonding Driver 214

List of Tables

Table 1: Typographical Conventions 13
Table 2: Abbreviations and Acronyms 13
Table 3: Clossary 15
Table 4: Reference Documents 16
Table 5: `mlnxofedinstall` Return Codes 29
Table 6: mlx4_vnic.conf file format 77
Table 7: Red Hat Linux mlx4_vnic.conf file format 78
Table 8: Supported ConnectX Port Configurations 101
Table 9: Recommended PCIe Configuration 106
Table 10: Useful MPI Links 113
Table 11: Congestion Control Manager General Options File 165
Table 12: Congestion Control Manager Switch Options File 165
Table 13: Congestion Control Manager CA Options File 165
Table 14: Congestion Control Manager CC MGR Options File 166
Table 15: ibdiagnet (of ibutils2) Output Files 170
Table 16: ibdiagnet (of ibutils) Output Files 173
Table 17: ibdiagpath Output Files 175
Table 18: ibv_devinfo Flags and Options 177
Table 19: ibstatus Flags and Options 179
Table 20: ibportstate Flags and Options 181
Table 21: ibportstate Flags and Options 185
Table 22: smpquery Flags and Options 189
Table 23: perfquery Flags and Options 193
Table 24: ibcheckerrs Flags and Options 196
Table 25: mstflint Switches 199
Table 26: mstflint Commands 200
Table 27: ibdump Options 203

Document Revision History

Printed on June 26, 2013.

Table 1 - Document Revision History

Document Revision	Date	Description
1.5.3-3.1.0	June 2013	Updated the “mlnx_add_kernel_support.sh” script options in Section 2.3.1, “Pre-installation Notes,” on page 23 Updated Section 6.2.4, “IRQ Affinity,” on page 99
1.5.3-3.1.0	September 2012	Updated Section 5.1, “Port Type Management,” on page 92
1.5.3-3.1.0	July 2012	Updated Section 4.6.3.2, “Static IPoIB Configuration,” on page 81
1.5.3-3.0.0	February 2012	Removed FCoE section
1.5.3-3.0.0	December 2011	• Updated Table 1, “mlnxofedinstall Return Codes,” on page 25 • Updated Table A.1.1, “Supported Mellanox Adapter Devices and Firmware,” on page 193 • Updated the installation Script in Section 2.3.3, “Installation Procedure,” on page 26 • Removed section “Socket Acceleration” • Added Section 4.5.4.3, “Para-Virtualized vNic,” on page 76 and its subsections • Added Section 4.5.3.7, “ALL VLAN,” on page 72 and its subsections • Updated sections “mlx4_core Parameters” “mlx4_en Parameters” on pages 222 and 223 • Updated Section 2.1.1, “Hardware Requirements,” on page 21 • Added new options to the Installation Script section on page 24 • Added Section 5.2, “Auto Sensing,” on page 93
1.5.3-1.0.0	July 2011	• Added Section 4.9, “Huge Pages Support for Queue Resources,” on page 90 • Updated Section 9.9, “Congestion Control,” on page 153 • Added Section 9.5.7.6, “Torus-2QoS Configuration File Syntax,” on page 134 • Updated Section A, “Mellanox FlexBoot,” on page 193

About this Manual

This Preface provides general information concerning the scope and organization of this User's Manual.

Intended Audience

This manual is intended for system administrators responsible for the installation, configuration, management and maintenance of the software and hardware of VPI (InfiniBand, Ethernet) adapter cards. It is also intended for application developers.

Common Abbreviations and Acronyms

Table 2 - Abbreviations and Acronyms (Sheet 1 of 2)

Abbreviation / Acronym	Whole Word / Description
B	(Capital) 'B' is used to indicate size in bytes or multiples of bytes (e.g., 1KB = 1024 bytes, and 1MB = 1048576 bytes)
b	(Small) 'b' is used to indicate size in bits or multiples of bits (e.g., 1Kb = 1024 bits)
FW	Firmware
HCA	Host Channel Adapter
HW	Hardware
IB	InfiniBand
LSB	Least significant *byte*
lsb	Least significant *bit*
MSB	Most significant *byte*
msb	Most significant *bit*
NIC	Network Interface Card
SW	Software
VPI	Virtual Protocol Interconnect
IPoIB	IP over InfiniBand
PFC	Priority Flow Control
PR	Path Record
RDS	Reliable Datagram Sockets
RoCE	RDMA over Converged Ethernet
SDP	Sockets Direct Protocol
SL	Service Level

Table 2 - Abbreviations and Acronyms (Sheet 2 of 2)

Abbreviation / Acronym	Whole Word / Description
SRP	SCSI RDMA Protocol
MPI	Message Passing Interface
EoIB	Ethernet over Infiniband
QoS	Quality of Service
ULP	Upper Level Protocol
VL	Virtual Lane
vHBA	Virtual SCSI Host Bus adapter
uDAPL	User Direct Access Programming Library

Glossary

The following is a list of concepts and terms related to InfiniBand in general and to Subnet Managers in particular. It is included here for ease of reference, but the main reference remains the *InfiniBand Architecture Specification*.

Table 3 - Glossary (Sheet 1 of 2)

Channel Adapter (CA), Host Channel Adapter (HCA)	An IB device that terminates an IB link and executes transport functions. This may be an HCA (Host CA) or a TCA (Target CA).
HCA Card	A network adapter card based on an InfiniBand channel adapter device.
IB Devices	Integrated circuit implementing InfiniBand compliant communication.
IB Cluster/Fabric/Subnet	A set of IB devices connected by IB cables.
In-Band	A term assigned to administration activities traversing the IB connectivity only.
Local Identifier (ID)	An address assigned to a port (data sink or source point) by the Subnet Manager, unique within the subnet, used for directing packets within the subnet.
Local Device/Node/System	The IB Host Channel Adapter (HCA) Card installed on the machine running IBDIAG tools.
Local Port	The IB port of the HCA through which IBDIAG tools connect to the IB fabric.
Master Subnet Manager	The Subnet Manager that is authoritative, that has the reference configuration information for the subnet. See Subnet Manager.

Table 3 - Glossary (Sheet 2 of 2)

Multicast Forwarding Tables	A table that exists in every switch providing the list of ports to forward received multicast packet. The table is organized by MLID.
Network Interface Card (NIC)	A network adapter card that plugs into the PCI Express slot and provides one or more ports to an Ethernet network.
Standby Subnet Manager	A Subnet Manager that is currently quiescent, and not in the role of a Master Subnet Manager, by agency of the master SM. See Subnet Manager.
Subnet Administrator (SA)	An application (normally part of the Subnet Manager) that implements the interface for querying and manipulating subnet management data.
Subnet Manager (SM)	One of several entities involved in the configuration and control of the an IB fabric.
Unicast Linear Forwarding Tables (LFT)	A table that exists in every switch providing the port through which packets should be sent to each LID.
Virtual Protocol Interconnet (VPI)	A Mellanox Technologies technology that allows Mellanox channel adapter devices (ConnectX®) to simultaneously connect to an InfiniBand subnet and a 10GigE subnet (each subnet connects to one of the adpater ports)

Related Documentation

Table 4 - Reference Documents

Document Name	Description
InfiniBand Architecture Specification, Vol. 1, Release 1.2.1	The InfiniBand Architecture Specification that is provided by IBTA
IEEE Std 802.3ae™-2002 (Amendment to IEEE Std 802.3-2002) Document # PDF: SS94996	Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications Amendment: Media Access Control (MAC) Parameters, Physical Layers, and Management Parameters for 10 Gb/s Operation
Firmware Release Notes for Mellanox adapter devices	See the Release Notes PDF file relevant to your adapter device under `docs/` folder of installed package.
MFT User's Manual	Mellanox Firmware Tools User's Manual. See under `docs/` folder of installed package.
MFT Release Notes	Release Notes for the Mellanox Firmware Tools. See under `docs/` folder of installed package.

Support and Updates Webpage

Please visit http://www.mellanox.com > Products > IB/VPI SW/Drivers for downloads, FAQ, troubleshooting, future updates to this manual, etc.

1 Mellanox OFED Overview

1.1 Introduction to Mellanox OFED

Mellanox OFED is a single Virtual Protocol Internconnect (VPI) software stack based on the OpenFabrics Enterprise Distribution (OFED) Linux stack, and operates across all Mellanox network adapter solutions supporting 10, 20, 40 and 56 Gb/s InfiniBand (IB); 10Gb/s and 40Gb/s Ethernet; and 2.5 or 5.0 GT/s PCI Express 2.0 and 8 GT/s PCI Express 3.0 uplinks to servers.

All Mellanox network adapter cards are compatible with OpenFabrics-based RDMA protocols and software, and are supported with major operating system distributions.

Mellanox OFED is certified with the following products:

- Mellanox Messaging Accelerator™ (VMA™) software: Multicast socket acceleration library that performs OS bypass for standard socket based applications.
- Mellanox Unified Fabric Manager™ (UFM™) software: Powerful platform for managing demanding scale-out computing fabric environments, built on top of the OpenSM industry standard routing engine.
- Fabric Collective Accelerator (FCA) - FCA is a Mellanox MPI-integrated software package that utilizes CORE-Direct technology for implementing the MPI collectives communications.

1.2 Introduction to Mellanox VPI Adapters

Mellanox VPI adapters, which are based on Mellanox ConnectX® family adapter devices, provide leading server and storage I/O performance with flexibility to support the myriad of communication protocols and network fabrics over a single device, without sacrificing functionality when consolidating I/O. For example, VPI-enabled adapters can support:

- Connectivity to 10, 20, 40 and 56Gb/s InfiniBand switches, Ethernet switches, emerging Data Center Ethernet switches
- A single firmware image for dual-port ConnectX/ConnectX-2/ConnectX-3 adapters that supports independent access to different convergence networks (InfiniBand, Ethernet or Data Center Ethernet) per port
- A unified application programming interface with access to communication protocols including: Networking (TCP, IP, UDP, sockets), Storage (NFS, CIFS, iSCSI, SRP, Clustered Storage), Clustering (MPI, DAPL, RDS, sockets), and Management (SNMP, SMI-S)
- Communication protocol acceleration engines including: networking, storage, clustering, virtualization and RDMA with enhanced quality of service
- RDMA over Converged Ethernet (RoCE). The following ULPs can be used over RoCE: uDAPL, SDP, RDS, MPI

1.3 Mellanox OFED Package

1.3.1 ISO Image

Mellanox OFED for Linux (MLNX_OFED_LINUX) is provided as ISO images, one per supported Linux distribution and CPU architecture, that includes *source code* and *binary* RPMs,

firmware, utilities, and documentation. The ISO image contains an installation script (called `mlnxofedinstall`) that performs the necessary steps to accomplish the following:

- Discover the currently installed kernel
- Uninstall any InfiniBand stacks that are part of the standard operating system distribution or another vendor's commercial stack
- Install the MLNX_OFED_LINUX binary RPMs (if they are available for the current kernel)
- Identify the currently installed InfiniBand HCAs and perform the required firmware updates

1.3.2 Software Components

MLNX_OFED_LINUX contains the following software components:

- Mellanox Host Channel Adapter Drivers
  - mthca (IB only)
  - mlx4 (VPI), which is split into multiple modules:
    - mlx4_core (low-level helper)
    - mlx4_ib (IB)
    - mlx4_en (Ethernet)
    - mlx4_vnic (EoIB)
- Mid-layer core
  - Verbs, MADs, SA, CM, CMA, uVerbs, uMADs
- Upper Layer Protocols (ULPs)
  - IPoIB, RDS, SDP, SRP Initiator, iSER
- MPI
  - Open MPI stack supporting the InfiniBand, RoCE and Ethernet interfaces
  - OSU MVAPICH stack supporting the InfiniBand and RoCE interfaces
  - MPI benchmark tests (OSU BW/LAT, Intel MPI Benchmark, Presta)
- OpenSM: InfiniBand Subnet Manager
- Utilities
  - Diagnostic tools
  - Performance tests
- Firmware tools (MFT)
- Source code for all the OFED software modules (for use under the conditions mentioned in the modules' LICENSE files)
- QIB
  - Low level driver implementation for all QLogic InfiniPath PCI-Express HCAs

    This driver was not tested by Mellanox Technologies.
- CXGB3
  - Provide RDMA and NIC support for the Chelsio S series adapters

    This driver was not tested by Mellanox Technologies.

- NES
  - Support for the NetEffect Ethernet Cluster Server Adapters

    This driver was not tested by Mellanox Technologies.
- Documentation

1.3.3 Firmware

The ISO image includes the following firmware items:

- Firmware images (.mlx format) for ConnectX® and ConnectX®-2 network adapters
- Firmware configuration (.INI) files for Mellanox standard network adapter cards and custom cards
- FlexBoot for ConnectX®, ConnectX®-2 HCA devices
- ConnectX EN PXE (gPXE boot) for ConnectX® EN and ConnectX®-2 EN devices

1.3.4 Directory Structure

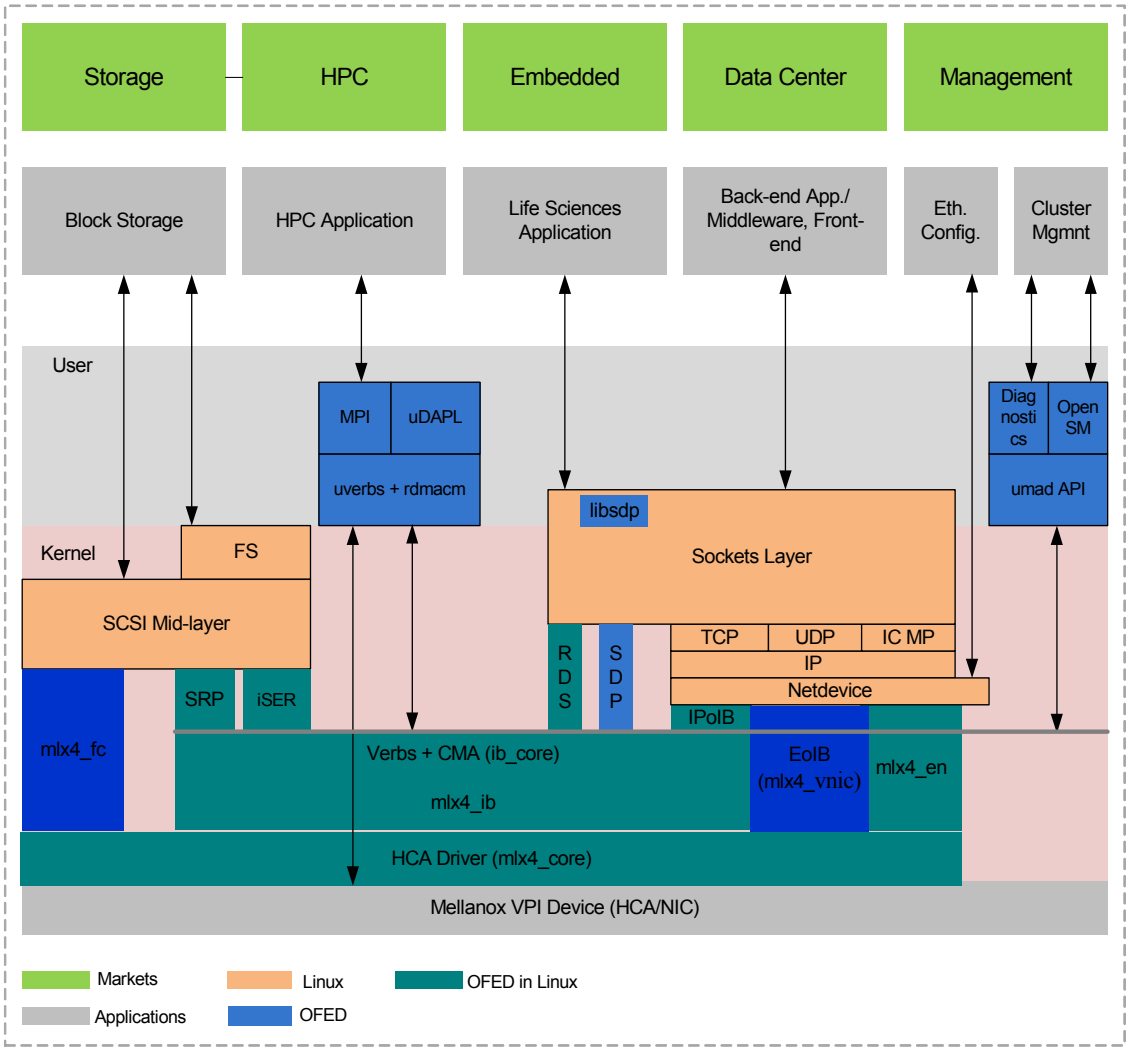
The ISO image of MLNX_OFED_LINUX contains the following files and directories:

- mlnxofedinstall - This is the MLNX_OFED_LINUX installation script.
- uninstall.sh - This is the MLNX_OFED_LINUX un-installation script.
- <RPMS folders> - Directory of binary RPMs for a specific CPU architecture.
- firmware/ - Directory of the Mellanox IB HCA firmware images (including Boot-over-IB)
- src/ - Directory of the OFED source tarball and the Mellanox Firmware Tools (MFT) tarball
- docs/ - Directory of Mellanox OFED related documentation

1.4 Architecture

Figure 1 shows a diagram of the Mellanox OFED stack, and how upper layer protocols (ULPs) interface with the hardware and with the kernel and user space. The application level also shows the versatility of markets that Mellanox OFED applies to.

Figure 1: Mellanox OFED Stack



The following sub-sections briefly describe the various components of the Mellanox OFED stack.

1.4.1 mthca HCA (IB) Driver

mthca is the low level driver implementation for the following Mellanox Technologies HCA (InfiniBand) devices: InfiniHost, InfiniHost III Ex and InfiniHost III Lx.

1.4.2 mlx4 VPI Driver

`mlx4` is the low level driver implementation for the ConnectX® family adapters designed by Mellanox Technologies. ConnectX family adapters can operate as an InfiniBand adapter, or as an Ethernet NIC. The OFED driver supports InfiniBand and Ethernet NIC configurations. To accommodate the supported configurations, the driver is split into four modules:

mlx4_core

Handles low-level functions like device initialization and firmware commands processing. Also controls resource allocation so that the InfiniBand and Ethernet functions can share the device without interfering with each other.

mlx4_ib

Handles InfiniBand-specific functions and plugs into the InfiniBand midlayer

mlx4_en

A 10/40GigE driver under drivers/net/mlx4 that handles Ethernet specific functions and plugs into the netdev mid-layer

1.4.3 Mid-layer Core

Core services include: management interface (MAD), connection manager (CM) interface, and Subnet Administrator (SA) interface. The stack includes components for both user-mode and kernel applications. The core services run in the kernel and expose an interface to user-mode for verbs, CM and management.

1.4.4 ULPs

IPoIB

The IP over IB (IPoIB) driver is a network interface implementation over InfiniBand. IPoIB encapsulates IP datagrams over an InfiniBand connected or datagram transport service. IPoIB pre-appends the IP datagrams with an encapsulation header, and sends the outcome over the InfiniBand transport service. The transport service is Reliable Connected (RC) by default, but it may also be configured to be Unreliable Datagram (UD). The interface supports unicast, multicast and broadcast. For details, see Chapter 4.6, “IP over InfiniBand”.

RoCE

RDMA over Converged Ethernet (RoCE) allows InfiniBand (IB) transport over Ethernet networks. It encapsulates IB transport and GRH headers in Ethernet packets bearing a dedicated ether type.

RDS

Reliable Datagram Sockets (RDS) is a socket API that provides reliable, in-order datagram delivery between sockets over RC or TCP/IP. For more details, see Chapter 4.2, “Reliable Datagram Sockets”.

SDP

Sockets Direct Protocol (SDP) is a byte-stream transport protocol that provides TCP stream semantics. SDP utilizes InfiniBand's advanced protocol offload capabilities. Because of this, SDP can have lower CPU and memory bandwidth utilization when compared to conventional implementations of TCP, while preserving the TCP APIs and semantics upon which most current network applications depend. For more details, see Chapter 4.3, "Sockets Direct Protocol".

SRP

SRP (SCSI RDMA Protocol) is designed to take full advantage of the protocol offload and RDMA features provided by the InfiniBand architecture. SRP allows a large body of SCSI software to be readily used on InfiniBand architecture. The SRP driver—known as the SRP Initiator—differs from traditional low-level SCSI drivers in Linux. The SRP Initiator does not control a local HBA; instead, it controls a connection to an I/O controller—known as the SRP Target—to provide access to remote storage devices across an InfiniBand fabric. The SRP Target resides in an I/O unit and provides storage services. See Chapter 4.4, "SCSI RDMA Protocol" and Appendix B, "SRP Target Driver".

uDAPL

User Direct Access Programming Library (uDAPL) is a standard API that promotes data center application data messaging performance, scalability, and reliability over RDMA interconnects: InfiniBand and RoCE. The uDAPL interface is defined by the DAT collaborative.

This release of the uDAPL reference implementation package for both DAT 1.2 and 2.0 specification is timed to coincide with OFED release of the Open Fabrics (www.openfabrics.org) software stack.

For more information about the DAT collaborative, go to the following site:

http://www.datcollaborative.org

1.4.5 MPI

Message Passing Interface (MPI) is a library specification that enables the development of parallel software libraries to utilize parallel computers, clusters, and heterogeneous networks. Mellanox OFED includes the following MPI implementations over InfiniBand:

- Open MPI – an open source MPI-2 implementation by the Open MPI Project
- OSU MVAPICH – an MPI-1 implementation by Ohio State University

Mellanox OFED also includes MPI benchmark tests such as OSU BW/LAT, Intel MPI Benchmark, and Presta.

1.4.6 InfiniBand Subnet Manager

All InfiniBand-compliant ULPs require a proper operation of a Subnet Manager (SM) running on the InfiniBand fabric, at all times. An SM can run on any node or on an IB switch. OpenSM is an InfiniBand-compliant Subnet Manager, and it is installed as part of Mellanox OFED.[1] See Chapter 9, "OpenSM – Subnet Manager".

1. OpenSM is disabled by default. See Chapter 9, "OpenSM – Subnet Manager" for details on enabling it.

1.4.7 Diagnostic Utilities

Mellanox OFED includes the following two diagnostic packages for use by network and data-center managers:

- ibutils – Mellanox Technologies diagnostic utilities
- infiniband-diags – OpenFabrics Alliance InfiniBand diagnostic tools

1.4.8 Mellanox Firmware Tools

The Mellanox Firmware Tools (MFT) package is a set of firmware management tools for a single InfiniBand node. MFT can be used for:

- Generating a standard or customized Mellanox firmware image
- Querying for firmware information
- Burning a firmware image to a single InfiniBand node

MFT includes the following tools:

`mlxburn`

This tool provides the following functions:

- Generation of a standard or customized Mellanox firmware image for burning—in .bin (binary) or .img format
- Burning an image to the Flash/EEPROM attached to a Mellanox HCA or switch device
- Querying the firmware version loaded on an HCA board
- Displaying the VPD (Vital Product Data) of an HCA board

`flint`

This tool burns a firmware binary image or an expansion ROM image to the Flash device of a Mellanox network adapter/bridge/switch device. It includes query functions to the burnt firmware image and to the binary image file.

`spark`

This tool burns a firmware binary image to the EEPROM(s) attached to an InfiniScaleIII® switch device. It includes query functions to the burnt firmware image and to the binary image file. The tool accesses the EEPROM and/or switch device via an I2C-compatible interface or via vendor-specific MADs over the InfiniBand fabric (In-Band tool).

`Debug utilities`

A set of debug utilities (e.g., itrace, mstdump, isw, and i2c)

For additional details, please refer to the MFT User's Manual `docs/`.

1.5 Quality of Service

Quality of Service (QoS) requirements stem from the realization of I/O consolidation over an IB network. As multiple applications and ULPs share the same fabric, a means is needed to control their use of network resources.

QoS over Mellanox OFED for Linux is discussed in Chapter 9, "OpenSM – Subnet Manager".

2 Installation

This chapter describes how to install and test the Mellanox OFED for Linux package on a single host machine with Mellanox InfiniBand and/or Ethernet adapter hardware installed. The chapter includes the following sections:

- Section 2.1,“Hardware and Software Requirements,” on page 21
- Section 2.2,“Downloading Mellanox OFED,” on page 22
- Section 2.3,“Installing Mellanox OFED,” on page 22
- Section 2.5,“Uninstalling Mellanox OFED,” on page 34

2.1 Hardware and Software Requirements

2.1.1 Hardware Requirements

Platforms

- A server platform with an adapter card based on one of the following Mellanox Technologies’ InfiniBand HCA devices:
  - MT25408 ConnectX®-2 (VPI, IB, EN) (firmware: fw-ConnectX2)
  - MT25408 ConnectX® (VPI, IB, EN) (firmware: fw-25408)
  - ConnectX®-3 (VPI, IB, EN) (firmware: fw-ConnectX3)

For the list of supported architecture platforms, please refer to the Mellanox OFED Release Notes file.

Required Disk Space for Installation

- 500 MB

Device ID

For the latest list of device IDs, please visit Mellanox website.

2.1.2 Software Requirements

Operating System

- Linux operating system

> For the list of supported operating system distributions and kernels, please refer to the Mellanox OFED Release Notes file.

Installer Privileges

- The installation requires administrator privileges on the target machine

2.2 Downloading Mellanox OFED

Step 1. Verify that the system has a Mellanox network adapter (HCA/NIC) installed by ensuring that you can see ConnectX or InfiniHost entries in the display.

The following example shows a system with an installed Mellanox HCA:

```
host1# lspci -v | grep Mellanox
02:00.0 InfiniBand: Mellanox Technologies MT25418 [ConnectX IB DDR, PCIe 2.0 2.5GT/s]
(rev a0)
```

Step 2. Download the ISO image to your host.

The image's name has the format MLNX_OFED_LINUX-<ver>-<OS label><CPU arch>.iso. You can download it from http://www.mellanox.com > Products > IB SW/Drivers.

Step 3. Use the md5sum utility to confirm the file integrity of your ISO image. Run the following command and compare the result to the value provided on the download page.

```
host1$ md5sum MLNX_OFED_LINUX-<ver>-<OS label>.iso
```

2.3 Installing Mellanox OFED

The installation script, `mlnxofedinstall`, performs the following:

- Discovers the currently installed kernel
- Uninstalls any software stacks that are part of the standard operating system distribution or another vendor's commercial stack
- Installs the MLNX_OFED_LINUX binary RPMs (if they are available for the current kernel)
- Identifies the currently installed InfiniBand and Ethernet network adapters and automatically[1] upgrades the firmware

1. The firmware will not be updated if you run the install script with the '--without-fw-update' option.

2.3.1 Pre-installation Notes

- The installation script removes all previously installed Mellanox OFED packages and re-installs from scratch. You will be prompted to acknowledge the deletion of the old packages.

  Pre-existing configuration files will be saved with the extension ".conf.saverpm".

- If you need to install Mellanox OFED on an entire (homogeneous) cluster, a common strategy is to mount the ISO image on one of the cluster nodes and then copy it to a shared file system such as NFS. To install on all the cluster nodes, use cluster-aware tools (such as pdsh).
- If your kernel version does not match with any of the offered pre-built RPMs, you can add your kernel version by using the "`mlnx_add_kernel_support.sh`" script located under the docs/ directory.

  Usage:

```
mlnx_add_kernel_support.sh -m|--mlnx_ofed <path to MLNX_OFED
>directory> [--make-iso|--make-tgz]
>
>[--make-iso]                    Create MLNX_OFED ISO image.
>[--make-tgz]                    Create MLNX_OFED tarball. (Default)
>[-t|--tmpdir <local work dir>]
>[--kmp]
>[-v|--verbose]
```

Example

The following command will create a MLNX_OFED_LINUX ISO image for RedHat 5.6 under the /tmp directory.

```
 MLNX_OFED_LINUX-1.5.3-rhel5.6-x86_64/docs/mlnx_add_kernel_support.sh -i /mnt/
MLNX_OFED_LINUX-1.5.3-rhel5.6-x86_64.iso
 All Mellanox, OEM, OFED, or Distribution IB packages will be removed.
 Do you want to continue?[y/N]:y
 Removing OFED RPMs...
 Running mkisofs...
 Created /tmp/MLNX_OFED_LINUX-1.5.3-rhel5.6-x86_64.iso
```

2.3.2 Installation Script

Mellanox OFED includes an installation script called `mlnxofedinstall`. Its usage is described below. You will use it during the installation procedure described in Section 2.3.3, "Installation Procedure," on page 26.

Usage

```
./mnt/mlnxofedinstall [OPTIONS]
```

Options

```
  -c|--   config <packages config_file>      Example of the configuration file can be
found
                                             under docs
  -n|--net <network config file>            Example of the network configuration file can
                                             be found under docs
  -p|--print-available                      Print available packages for the current plat
                                               form and create a corresponding ofed.conf
file.
                                            The installation script exits after creating
                                             ofed.conf.
  --without-32bit                            Skip 32-bit libraries installation
  --without-depcheck                         Skip Distro's libraries check
  --without-fw-update                        Skip firmware update
  --force-fw-update                          Force firmware update
  --force                                   Force installation (without querying the user)
  --all                                     Install all kernel modules, libibverbs, libibu
                                             mad,librdmacm, mft, mstflint,
                                                diagnostic tools, OpenSM, ib-bonding,
MVAPICH,
                                            Open MPI, MPI tests, MPI selector, perftest,
                                             sdpnetstat and libsdp srptools, rds tools,
                                             static and dynamic libraries
  --hpc                                      Install all kernel modules, libibverbs,
                                             libibumad,librdmacm, mft, mstflint,
                                           diagnostic tools, OpenSM, ib-bonding, MVAPICH,
                                             Open MPI, MPI tests, MPI selector, dynamic
                                             libraries
  --basic                                   Install all kernel modules, libibverbs, libibu
                                             mad, mft, mstflint, dynamic libraries
  --msm                                      Install all kernel modules, libibverbs,
                                            libibumad, mft, mstflint, diagnostic tools,
                                             OpenSM, ib-bonding, dynamic libraries
                                            NOTE: With --msm flag, the OpenSM daemon is
                                             configured to run upon boot.
  --vma                                      Install packages required by VMA to support
                                           both IB and Ethernet
  --vma-ib                                  Install packages required by VMA to work over
                                             InfiniBand
  --vma-eth                                 Install packages required by VMA to work over
                                             Ethernet
  -v|-vv|-vvv                                Set verbosity level
```

```
  --pfc <0|bitmask>                       Priority based Flow Control policy on TX and
RX
                                           [7:0]
  -q                                       Set quiet - no messages will be printed
  --umad-dev-rw                            Grant non root users read/write
                                           permission for umad devices instead of
                                           default
```

2.3.2.1 mlnxofedinstall Return Codes

Table 1 lists the `mlnxofedinstall` script return codes and their meanings.

Table 1 - mlnxofedinstall Return Codes

Return Code	Meaning
0	The Installation ended successfully
1	The installation failed
2	No firmware was found for the adapter device
22	Invalid parameter
28	Not enough free space
171	Not applicable to this system configuration. This can occur when the required hardware is not present on the system.
172	Prerequisites are not met. For example, missing the required software installed or the hardware is not configured correctly.
173	Failed to start the `mst` driver

2.3.3 Installation Procedure

Step 1. Login to the installation machine as root.

Step 2. Mount the ISO image on your machine

```
host1# mount -o ro,loop MLNX_OFED_LINUX-<ver>-<OS label>-<CPU arch>.iso /mnt
```

Step 3. Run the installation script.

```
./mlnxofedinstall
This program will install the MLNX_OFED_LINUX package on your machine.
Note that all other Mellanox, OEM, OFED, or Distribution IB packages will be removed.
Uninstalling the previous version of OFED

Starting MLNX_OFED_LINUX-1.5.3-1.0.5 installation ...

Installing mlnx-ofa_kernel RPM
Preparing...                ########################################################
mlnx-ofa_kernel             ########################################################
Installing kmod-mlnx-ofa_kernel RPM
Preparing...                ########################################################
kmod-mlnx-ofa_kernel        ########################################################
Installing mlnx-ofa_kernel-devel RPM
Preparing...                ########################################################
mlnx-ofa_kernel-devel       ########################################################
Installing kernel-mft RPM
Preparing...                ########################################################
kernel-mft                  ########################################################
Installing mlx4_accl_sys RPM
Preparing...                ########################################################
mlx4_accl_sys               ########################################################
Installing mlx4_accl RPM
Preparing...                ########################################################
mlx4_accl                   ########################################################
Installing mpi-selector RPM
Preparing...                ########################################################
mpi-selector                ########################################################
Install user level RPMs
Preparing...                ########################################################
libibverbs                  ########################################################
libibumad                   ########################################################
librdmacm                   ########################################################
opensm-libs                 ########################################################
libibmad                    ########################################################
libmverbs                   ########################################################
libmqe                      ########################################################
dapl                        ########################################################
ibutils2                    ########################################################
mvapich_gcc                 ########################################################
libmthca                    ########################################################
```

```
libmlx4                  ##################################################
libcxgb3                 ##################################################
libnes                   ##################################################
libipathverbs            ##################################################
libibcm                  ##################################################
infinipath-psm           ##################################################
openmpi_gcc              ##################################################
libsdp                   ##################################################
compat-dapl              ##################################################
mpitests_openmpi_gcc     ##################################################
mpitests_openmpi_intel   ##################################################
mpitests_mvapich_gcc     ##################################################
mpitests_mvapich_intel   ##################################################
cc_mgr                   ##################################################
ar_mgr                   ##################################################
dapl-utils               ##################################################
ibsim                    ##################################################
infiniband-diags         ##################################################
opensm                   ##################################################
ibutils                  ##################################################
librdmacm-utils          ##################################################
perftest                 ##################################################
qperf                    ##################################################
ibacm                    ##################################################
srptools                 ##################################################
mvapich_intel            ##################################################
libibverbs-utils         ##################################################
ibdump                   ##################################################
openmpi_intel            ##################################################
dump_pr                  ##################################################
rds-tools                ##################################################
sdpnetstat               ##################################################
mstflint                 ##################################################
libibumad-devel          ##################################################
libibverbs-devel         ##################################################
libibcm-devel            ##################################################
librdmacm-devel          ##################################################
libibmad-devel           ##################################################
```

```
opensm-devel              ##################################################
opensm-static             ##################################################
opensm-static             ##################################################
compat-dapl-devel         ##################################################
libsdp-devel              ##################################################
infinipath-psm-devel      ##################################################
libipathverbs-devel       ##################################################
libipathverbs-devel       ##################################################
libnes-devel-static       ##################################################
libnes-devel-static       ##################################################
libcxgb3-devel            ##################################################
libcxgb3-devel            ##################################################
libmlx4-devel             ##################################################
libmlx4-devel             ##################################################
libmthca-devel-static     ##################################################
libmthca-devel-static     ##################################################
dapl-devel                ##################################################
libmqe-devel              ##################################################
libmverbs-devel           ##################################################
```

```
  libibmad-static            ##################################################
  libibmad-static            ##################################################
  libibumad-static           ##################################################
  libibumad-static           ##################################################
  rds-devel                  ##################################################
  mft                        ##################################################
  dapl-devel-static          ##################################################
  dapl-devel-static          ##################################################
  libibverbs-devel-static    ##################################################
  libibverbs-devel-static    ##################################################
  mlnxofed-docs              ##################################################
  ofed-scripts               ##################################################
  libibverbs                 ##################################################
  librdmacm                  ##################################################
  libibumad                  ##################################################
  libmverbs                  ##################################################
  libmqe                     ##################################################
  libibmad                   ##################################################
  opensm-libs                ##################################################
  compat-dapl                ##################################################
  dapl                       ##################################################
  libibcm                    ##################################################
  libsdp                     ##################################################
  libmthca                   ##################################################
  libmlx4                    ##################################################
  libcxgb3                   ##################################################
  libnes                     ##################################################
  libipathverbs              ##################################################
  libsdp-devel               ##################################################
  libibcm-devel              ##################################################
  dapl-devel                 ##################################################
  compat-dapl-devel          ##################################################
  opensm-devel               ##################################################
  libibmad-devel             ##################################################
  libmqe-devel               ##################################################
  libmverbs-devel            ##################################################
  libibumad-devel            ##################################################
  librdmacm-devel            ##################################################
  libibverbs-devel           ##################################################
  Device (15b3:673c):
            02:00.0 InfiniBand: Mellanox Technologies MT26428 [ConnectX VPI PCIe 2.0 5GT/
s -
            IB QDR / 10GigE] (rev b0)
            Link Width: 8x
            PCI Link Speed: 2.5Gb/s
  Installation finished successfully
```

In case your machine has the latest firmware, no firmware update will occur and the installation script will print at the end of installation a message similar to the following:

```
...
Installation finished successfully.
The firmware version 2.9.1000 is up to date.
Note: To force firmware update use '--force-fw-update' flag
```

In case your machine has an unsupported network adapter device, no firmware update will occur and the error message below will be printed. Please contact your hardware vendor for help on firmware updates.

Error message:

```
-I- Querying device ...
-E- Can't auto detect fw configuration file: ...
```

Step 4. In case the installation script performed firmware updates to your network adapter hardware, it will ask you to reboot your machine.

Step 5. The script adds the following lines to `/etc/security/limits.conf` for the userspace components such as MPI:

```
* soft memlock unlimited
* hard memlock unlimited
```

These settings unlimit the amount of memory that can be pinned by a user space application.

If desired, tune the value unlimited to a specific amount of RAM.

Step 6. For your machine to be part of the InfiniBand/VPI fabric, a Subnet Manager must be running on one of the fabric nodes. At this point, Mellanox OFED for Linux has already installed the OpenSM Subnet Manager on your machine. For details on starting OpenSM, see Chapter 9, “OpenSM – Subnet Manager”.

Step 7. (InfiniBand only) Run the `hca_self_test.ofed` utility to verify whether or not the InfiniBand link is up. The utility also checks for and displays additional information such as

- HCA firmware version
- Kernel architecture
- Driver version
- Number of active HCA ports along with their states
- Node GUID

Note: For more details on `hca_self_test.ofed`, see the file `hca_self_test.readme` under `docs/`.

```
host1# /usr/bin/hca_self_test.ofed
----
Performing InfiniBand HCA Self Test ----
Number of HCAs Detected ..... 1
PCI Device Check ..... PASS
Kernel Arch ..... x86_64
Host Driver Version ..... MLNX_OFED_LINUX-1.5.3 (OFED-1.5.3): 1.5.2-
2.6.32.12_0.7_default
Host Driver RPM Check ..... PASS
HCA Firmware on HCA #0 ..... v2.9.1000
HCA Firmware Check on HCA #0 ..... PASS
Host Driver Initialization ..... PASS
Number of HCA Ports Active ..... 1
```

```
Port State of Port #1 on HCA #0 ........ UP 4X DDR
Port State of Port #2 on HCA #0 ........ INIT
Error Counter Check on HCA #0 .......... PASS
Kernel Syslog Check .................... PASS
Node GUID on HCA #0 .................... 00:02:c9:03:00:00:10:e0
------------------ DONE --------------------
```

After the installer completes, information about the Mellanox OFED installation such as prefix, kernel version, and installation parameters can be retrieved by running the command **/etc/infiniband/info**.

2.3.4 Installation Results

Software

- The OFED and MFT packages are installed under the /usr directory.
- The kernel modules are installed under:
  - InfiniBand subsystem:

```
/lib/modules/`uname -r`/updates/kernel/drivers/infiniband/
```

  - mlx4 driver:

  Under **/lib/modules/`uname -r`/updates/kernel/drivers/net/mlx4** you will find **mlx4_core.ko**, **mlx4_en.ko**, **mlx4_ib.ko**, **mlx4_vnic.ko** and **mlx4_fc.ko**

  - IPoIB:

```
/lib/modules/`uname -r`/updates/kernel/drivers/infiniband/ulp/ipoib/ib_ipoib.ko
```

  - SDP:

```
/lib/modules/`uname -r`/updates/kernel/drivers/infiniband/ulp/sdp/ib_sdp.ko
```

  - SRP

```
/lib/modules/`uname -r`/updates/kernel/drivers/infiniband/ulp/srp/ib_srp.ko
```

  - RDS:

```
/lib/modules/`uname -r`/updates/kernel/net/rds/rds.ko
/lib/modules/`uname -r`/updates/kernel/net/rds/rds_rdma.ko
/lib/modules/`uname -r`/updates/kernel/net/rds/rds_tcp.ko
```

- The package kernel-ib-devel include files are placed under `/usr/src/ofa_kernel/include/`. These include files should be used when building kernel modules that use the stack. (Note that the include files, if needed, are “backported” to your kernel.)
- The raw package (un-backported) source files are placed under `/usr/src/ofa_kernel-<ver>`

- The script `openibd` is installed under `/etc/init.d/`. This script can be used to load and unload the software stack.
- The script `connectx_port_config` is installed under `/sbin`. This script can be used to configure the ports of ConnectX network adapter cards to Ethernet and/or InfiniBand. For details on this script, please see Section 5.1, "Port Type Management".
- The directory `/etc/infiniband` is created with the files `info` and `openib.conf` and `connectx.conf`. The `info` script can be used to retrieve Mellanox OFED installation information. The `openib.conf` file contains the list of modules that are loaded when the `openibd` script is used. The `connectx.conf` file saves the ConnectX adapter card's ports configuration to Ethernet and/or InfiniBand. This file is used at driver start/restart (`/etc/init.d/openibd start`)
- The file `90-ib.rules` is installed under `/etc/udev/rules.d/`
- If OpenSM is installed, the daemon `opensmd` is installed under `/etc/init.d/` and `opensm.conf` is installed under `/etc`.
- If IPoIB configuration files are included, `ifcfg-ib<n>` files will be installed under:
  - `/etc/sysconfig/network-scripts/` on a RedHat machine
  - `/etc/sysconfig/network/` on a SuSE machine
- The installation process unlimits the amount of memory that can be pinned by a user space application. See Step 5.
- Man pages will be installed under `/usr/share/man/`

Firmware

- The firmware of existing network adapter devices will be updated if the following two conditions are fullfilled:
  1. You run the installation script in default mode; that is, *without* the option '--without-fw-update'.
  2. The firmware version of the adapter device is older than the firmware version included with the Mellanox OFED ISO image

  If an adapter's Flash was originially programmed with an Expansion ROM image, the automatic firmware update will also burn an Expansion ROM image.
- In case your machine has an unsupported network adapter device, no firmware update will occur and the error message below will be printed. Please contact your hardware vendor for help on firmware updates.

  Error message:

```
-I- Querying device ...
-E- Can't auto detect fw configuration file: ...
```

2.3.5 Post-installation Notes

- Most of the Mellanox OFED components can be configured or reconfigured after the installation by modifying the relevant configuration files. See the relevant chapters in this manual for details.
- The list of the modules that will be loaded automatically upon boot can be found in the `/etc/infiniband/openib.conf` file.

2.4 Updating Firmware After Installation

In case you ran the `mlnxofedinstall` script with the '`--without-fw-update`' option and now you wish to (manually) update firmware on your adapter card(s), you need to perform the following steps:

> If you need to burn an Expansion ROM image, please refer to "Burning the Expansion ROM Image" on page 194

> The following steps are also appropriate in case you wish to burn newer firmware that you have downloaded from Mellanox Technologies' Web site (http://www.mellanox.com > Downloads > Firmware).

Step 1. Start `mst`.

```
host1# mst start
```

Step 2. Identify your target InfiniBand device for firmware update.

3. Get the list of InfiniBand device names on your machine.

```
host1# mst status
MST modules:
------------
    MST PCI module loaded
    MST PCI configuration module loaded
    MST Calibre (I2C) module is not loaded

MST devices:
------------
/dev/mst/mt25418_pciconf0        - PCI configuration cycles access.
                                   bus:dev.fn=02:00.0 addr.reg=88
                                   data.reg=92
                                   Chip revision is: A0
/dev/mst/mt25418_pci_cr0         - PCI direct access.
                                   bus:dev.fn=02:00.0 bar=0xdef00000
                                   size=0x100000
```

```
                                    Chip revision is: A0
 /dev/mst/mt25418_pci_msix0       - PCI direct access.
                                    bus:dev.fn=02:00.0 bar=0xdeefe000
                                    size=0x2000
 /dev/mst/mt25418_pci_uar0        - PCI direct access.
                                    bus:dev.fn=02:00.0 bar=0xdc800000
                                    size=0x800000
```

4. Your InfiniBand device is the one with the postfix "_pci_cr0". In the example listed above, this will be `/dev/mst/mt25418_pci_cr0`.

Step 3. Burn firmware.

1. Burning a firmware binary image using **`mstflint`** (that is already installed on your machine). Please refer to `MSTFLINT_README.txt` under `docs/`.
2. Burning a firmware image from a .mlx file using the **`mlxburn`** utility (that is already installed on your machine).

The following command burns firmware onto the ConnectX device with the device name obtained in the example of Step 2.

```
 host1$ mlxburn -dev /dev/mst/mt25418_pci_cr0 -fw /mnt/firmware/fw-25408/fw-25408-
rel.mlx
```

Step 4. Reboot your machine after the firmware burning is completed.

2.5 Uninstalling Mellanox OFED

Use the script `/usr/sbin/ofed_uninstall.sh` to uninstall the Mellanox OFED package. The script is part of the `ofed-scripts` RPM.

3 Configuration Files

For the complete list of configuration files, please refer to MLNX_OFED_configuration_files.txt

4 Driver Features

4.1 RDMA over Converged Ethernet

4.1.1 RoCE Overview

RDMA over Converged Ethernet (RoCE) allows InfiniBand (IB) transport over Ethernet networks. It encapsulates IB transport and GRH headers in Ethernet packets bearing a dedicated ether type.

While the use of GRH is optional within IB subnets, it is mandatory when using RoCE. Verbs applications written over IB verbs should work seamlessly, but they require provisioning of GRH information when creating address vectors. The library and driver are modified to provide for mapping from GID to MAC addresses required by the hardware.

4.1.2 Software Dependencies

In order to use RoCE over Mellanox ConnectX(R) hardware, the mlx4_en driver must be loaded. Please refer to MLNX_EN_README.txt for further details.

4.1.3 Firmware Dependencies

In order to use RoCE over Mellanox ConnectX(R) hardware, RoCE requires ConnectX® firmware version 2.7.000 or higher. Features such as loopback require higher firmware versions.

4.1.4 General Guidelines

Since RoCE encapsulates InfiniBand traffic in Ethernet frames, the corresponding net device must be up and running. In case of Mellanox hardware, mlx4_en must be loaded and the corresponding interface configured.

- Make sure that `mlx4_en.ko` is loaded. To verify the module is loaded, run the folowing command: “lsmod | grep mlx4_en. If the module is loaded, the mlnx4_en should be displayed as shown in the example below.

```
# lsmod | grep mlx4_en
*mlx4_en 75276 0
```

- Run “ibv_devinfo”. There is a new field named “link_layer” which can be either “Ethernet” or “IB”. If the value is IB, then you need to use connectx_port_config to change the ConnectX family adapters ports designation to eth (see mlx4_release_notes.txt for details)
- Configure the IP address of the interface so that the link will become active
- All IB verbs applications which run over IB verbs should work on RoCE links as long as they use GRH headers (that is, as long as they specify use of GRH in their address vector)

4.1.5 Ported Applications

The following applications are ported with RoCE:

- ibv_*_pingpong examples are ported. The user must specify the GID of the remote peer using the new '-g' option. The GID has the same format as that in /sys/class/infiniband/mlx4_0/ports/1/gids/0

Care should be taken when using ibv_ud_pingpong. The default message size is 2K, which is likely to exceed the MTU of the RoCE link. Use ibv_devinfo to inspect the link MTU and specify an appropriate message size.

- All rdma_cm applications should work seamlessly without any change
- libsdp works without any change
- Performance tests

4.1.6 GID Tables

With RoCE, there may be several entries in a port's GID table. The first entry always contains the IPv6 link's local address of the corresponding Ethernet interface. The link's local address is formed in the following way:

```
gid[0..7] = fe80000000000000
gid[8] = mac[0] ^ 2
gid[9] = mac[1]
gid[10] = mac[2]
gid[11] = ff
gid[12] = fe
gid[13] = mac[3]
gid[14] = mac[4]
gid[15] = mac[5]
```

If VLAN is supported by the kernel and there are VLAN interfaces on the main Ethernet interface (the interface that the IB port is tied to), then each such VLAN will appear as a new GID in the port's GID table. The format of the GID entry will be identical to the one described above, except for the following change:

```
gid[11] = VLAN ID high byte (4 MS bits).
gid[12] = VLAN ID low byte
```

Please note that VLAN ID is 12 bits wide.

4.1.6.1 Priority Pause Frames

Tagged Ethernet frames carry a 3-bit priority field. The value of this field is derived from the IB SL field by taking the 3 least significant bits of the SL field.

4.1.7 Using VLANs

In order for RoCE traffic to use VLAN tagged frames, the user needs to specify GID table entries that are derived from VLAN devices when creating address vectors. Consider the example below.

- Make sure VLAN support is enabled by the kernel. Usually this requires loading the 802.1q module.

  > modprobe 8021q

- Add a VLAN device

  > vconfig add eth2 7

- Assign an IP address to the VLAN interface. This should create a new entry in the GID table (as index 1)

  > ifconfig eth2.7 7.10.11.12

- Verbs test

  On server: > ibv_rc_pingpong -g 1

  On client: > ibv_rc_pingpongs -g 1 server

- For rdma_cm applications, the user needs only to specify an IP address of a VLAN device for the traffic to go with the VLAN tagged frames.

4.1.8 Reading Port Counters Statistics

It is possible to read port statistics in the same way it is done for regular InfiniBand ports. The information is available from the sysfs at `/sys/class/infiniband/<device>/ports/<port number>/counters`, and the supported counters are `port_rcv_packets`, `port_xmit_packets`, `port_rcv_data` and `port_xmit_data`. These counters count InfiniBand data only, and do not account for Ethernet traffic.

For example, to read the number of transmited packets, run:

```
> cat /sys/class/infiniband/<device>/ports/<port number>/counters/port_xmit_packets
```

RoCE traffic is not shown in the associated Etherent device's counters since it is offloaded by the hardware and does not go through Ethernet network driver.

4.1.9 A Detailed Example

This section provides a step-by-step example of using InfiniBand over Ethernet (RoCE).

Installation and Driver Loading

The MLNX OFED installation script installs RoCE as part of mlx4 and mlx4_en and other modules. See Section 2.3, "Installing Mellanox OFED" for details on installation.

The list of the modules that will be loaded automatically upon boot can be found in the configuration file /etc/infiniband/openib.conf.

Enter the following command to display the current run of MLNX OFED.

```
# ibv_devinfo
hca_id: mlx4_0
        transport: InfiniBand (0)
        fw_ver: 2.7.700
        node_guid: 0002:c903:0008:e810
        sys_image_guid: 0002:c903:0008:e813
        vendor_id: 0x02c9
        vendor_part_id: 26428
        hw_ver: 0xB0
        board_id: MT_0DD0120009
        phys_port_cnt: 2
             port: 1
                   state: PORT_INIT (2)
                   max_mtu: 2048 (4)
                   active_mtu: 2048 (4)
                   sm_lid: 0
                   port_lid: 0
                   port_lmc: 0x00
                   link_layer: IB
             port: 2
                   state: PORT_ACTIVE (4)
                   max_mtu: 2048 (4)
                   active_mtu: 1024 (3)
                   sm_lid: 0
                   port_lid: 0
                   port_lmc: 0x00
                   link_layer: Ethernet
#
```

Notes regarding the command output:

1. The InfiniBand port (port 1) is in PORT_INIT state, and the Ethernet port (port 2) is in PORT_ACTIVE state. You can also run the following commands to obtain the port state.

```
# cat /sys/class/infiniband/mlx4_0/ports/1/state
2: INIT
# cat /sys/class/infiniband/mlx4_0/ports/2/state
4: ACTIVE
#
```

2. Look at the link_layer parameter of each port. In this case port 1 is IB and port 2 is Ethernet. Nevertheless, port 2 appears in the list of the HCA's ports. You can also run the following commands to obtain the link_layer of the two ports:

```
# cat /sys/class/infiniband/mlx4_0/ports/1/link_layer
InfiniBand
# cat /sys/class/infiniband/mlx4_0/ports/2/link_layer
Ethernet
#
```

3. The firmware version is 2.7.700 (appears at the top). You can also run the following command to obtain the firmware version:

```
# cat /sys/class/infiniband/mlx4_0/fw_ver
2.7.700
#
```

4. The IB over Ethernet's Port MTU is 2K byte at maximum, however the actual MTU cannot exceed the mlx4_en interface's MTU. Since the mlx4_en interface's MTU is 1560, port 2 will run with MTU of 1K. Please note that RoCE's MTU are subject to IB MTU restrictions. The RoCE's MTU values are, 256 byte, 512 byte, 1024 byte and 2K.

Association of IB Ports to Ethernet Ports

It is useful to know how IB ports associate to network ports.

```
# ibdev2netdev
mlx4_0 port 2 <===> eth2
mlx4_0 port 1 <===> ib0
#
```

Since both RoCE and mlx4_en use the Ethernet port of the adapter, one of the drivers must carry the task of controlling the port state. In this implementation, it is the task of the mlx4_en driver. The mlx4_ib driver holds a reference to the mlx4_en net device for getting notifications about the state of the port, as well as using the mlx4_en driver to resolve IP addresses to MAC that are required for address vector creation. However, RoCE traffic does not go through the mlx4_en driver; it is completely offloaded by the hardware.

Configre an IP Address to mlx4_en Interface

Run the following on both sides of the link.

```
# ifconfig eth2 20.4.3.220
# ifconfig eth2
eth2              Link encap:Ethernet HWaddr 00:02:C9:08:E8:11
                  inet addr:20.4.3.220 Bcast:20.255.255.255 Mask:255.0.0.0
                  UP BROADCAST MULTICAST MTU:1500 Metric:1
                  RX packets:0 errors:0 dropped:0 overruns:0 frame:0
                  TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
                  collisions:0 txqueuelen:1000
                  RX bytes:0 (0.0 b) TX bytes:0 (0.0 b)
```

Make sure that ping is working:

```
# ping 20.4.3.219
PING 20.4.3.219 (20.4.3.219) 56(84) bytes of data.
64 bytes from 20.4.3.219: icmp_seq=1 ttl=64 time=0.873 ms
64 bytes from 20.4.3.219: icmp_seq=2 ttl=64 time=0.198 ms
64 bytes from 20.4.3.219: icmp_seq=3 ttl=64 time=0.167 ms
--- 20.4.3.219 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.167/0.412/0.873/0.326 ms
```

Inspecting the GID Table

```
# cat /sys/class/infiniband/mlx4_0/ports/2/gids/0
fe80:0000:0000:0000:0202:c9ff:fe08:e811
#
# cat /sys/class/infiniband/mlx4_0/ports/2/gids/1
0000:0000:0000:0000:0000:0000:0000:0000
#
```

According to the output, we currently have one entry only.

Run an Example Test - ibv_rc_pingpong

Start the server first:

```
# ibv_rc_pingpong -g 0 -i 2
local address: LID 0x0000, QPN 0x00004f, PSN 0x3315f6, GID fe80::202:c9ff:fe08:e799
remote address: LID 0x0000, QPN 0x04004f, PSN 0x2cdede, GID fe80::202:c9ff:fe08:e811
8192000 bytes in 0.01 seconds = 4730.13 Mbit/sec
1000 iters in 0.01 seconds = 13.85 usec/iter
#
```

Then start the client:

```
# ibv_rc_pingpong -g 0 -i 2 sw419
local address: LID 0x0000, QPN 0x04004f, PSN 0x2cdede, GID fe80::202:c9ff:fe08:e811
remote address: LID 0x0000, QPN 0x00004f, PSN 0x3315f6, GID fe80::202:c9ff:fe08:e799
8192000 bytes in 0.01 seconds = 4787.84 Mbit/sec
1000 iters in 0.01 seconds = 13.69 usec/iter
#
```

Add VLANs

Make sure that the 8021.q module is loaded:

```
# modprobe 8021q
```

Add the VLAN device:

```
# vconfig add eth2 7
Added VLAN with VID == 7 to IF -:eth2:-
#
```

Configure an IP address for it:

```
# ifconfig eth2.7 7.4.3.220
```

Examine the GID table:

```
# cat /sys/class/infiniband/mlx4_0/ports/2/gids/0
fe80:0000:0000:0000:0202:c9ff:fe08:e811
#
# cat /sys/class/infiniband/mlx4_0/ports/2/gids/1
fe80:0000:0000:0000:0202:c900:0708:e811
```

According to the output, we now have two entries.

Run the Example Again, Now on VLAN

On Server:

```
# ibv_rc_pingpong -g 1 -i 2
local address: LID 0x0000, QPN 0x04004f, PSN 0xbdde2c, GID fe80::202:c900:708:e799
remote address: LID 0x0000, QPN 0x08004f, PSN 0xc9d800, GID fe80::202:c900:708:e811
8192000 bytes in 0.01 seconds = 4824.50 Mbit/sec
1000 iters in 0.01 seconds = 13.58 usec/iter
```

On Client:

```
# ibv_rc_pingpong -g 1 -i 2 sw419
local address: LID 0x0000, QPN 0x08004f, PSN 0xc9d800, GID fe80::202:c900:708:e811
remote address: LID 0x0000, QPN 0x04004f, PSN 0xbdde2c, GID fe80::202:c900:708:e799
8192000 bytes in 0.01 seconds = 4844.83 Mbit/sec
1000 iters in 0.01 seconds = 13.53 usec/iter
```

Defining Ethernet Priority (PCP in 802.1q Headers)

On Server:

```
# ibv_rc_pingpong -g 1 -i 2 -l 4
local address: LID 0x0000, QPN 0x1c004f, PSN 0x9daf6c, GID fe80::202:c900:708:e799
remote address: LID 0x0000, QPN 0x1c004f, PSN 0xb0a49b, GID fe80::202:c900:708:e811
8192000 bytes in 0.01 seconds = 4840.89 Mbit/sec
1000 iters in 0.01 seconds = 13.54 usec/iter
```

On Client:

```
# ibv_rc_pingpong -g 1 -i 2 -l 4 sw419
local address: LID 0x0000, QPN 0x1c004f, PSN 0xb0a49b, GID fe80::202:c900:708:e811
remote address: LID 0x0000, QPN 0x1c004f, PSN 0x9daf6c, GID fe80::202:c900:708:e799
8192000 bytes in 0.01 seconds = 4855.96 Mbit/sec
1000 iters in 0.01 seconds = 13.50 usec/iter
```

Using rdma_cm Tests

On Server:

```
# ucmatose
cmatose: starting server
initiating data transfers
completing sends
receiving data transfers
data transfers complete
cmatose: disconnecting
disconnected
test complete
return status 0
#
```

On Client:

```
# ucmatose -s 20.4.3.219
cmatose: starting client
cmatose: connecting
receiving data transfers
sending replies
data transfers complete
test complete
return status 0
#
```

This server-client run is without PCP or VLAN because the IP address used does not belong to a VLAN interface. If you specify a VLAN IP address, then traffic should go over VLAN.

Type Of Service (TOS)

The TOS field for rdma_cm sockets can be set using the rdma_set_option() API, just as it is set for regular sockets. If the user does not set a TOS, the default value (0) will be used. Within the rdma_cm kernel driver, the TOS field is converted into an SL field. The conversion formula is as follows:

SL = TOS >> 5 (e.g., take the 3 most significant bits of the TOS field)

In the hardware driver, the SL field is converted into PCP by the following formula:

PCP = SL & 7 (take the 3 least significant bits of the TOS field)

Note: SL affects the PCP only when the traffic goes over tagged VLAN frames.

4.1.10 Configuring DAPL over RoCE

The default dat.conf file which contains entries for the DAPL devices, does not contain entries for the DAPL over RDMA_CM over RoCE devices.

To add the missing entries perform the following:

Step 1. Run the ibdev2netdev utility to see all the associations between the Ethernet devices and the IB devices/ports.

Step 2. Add a new entry line according to the format below to the dat.conf file for each output line of the ibdev2netdev utility.

<IA Name> u2.0 nonthreadsafe default libdaplofa.so.2 dapl.2.0 "<ethX> <port>" ""

Parameter	Description	Example
<IA Name>	The device's IA name. The name must be unique.	ofa-v2-ethx
<ethX>	The associated Ethernet device used by RoCE.	eth3
<port>	The port number.	1

The following is an example of the ibdev2netdev utility's output and the entries added per each output line:

Example:

```
sw419:~ # ibdev2netdev
mlx4_0 port 2 <===> eth2
mlx4_0 port 1 <===> eth3
ofa-v2-eth2 u2.0 nonthreadsafe default libdaplofa.so.2 dapl.2.0 "eth2 2"
""
ofa-v2-eth3 u2.0 nonthreadsafe default libdaplofa.so.2 dapl.2.0 "eth3 1"
""
```

4.2 Reliable Datagram Sockets

4.2.1 Overview

Reliable Datagram Sockets (RDS) is a socket API that provides reliable, in-order datagram delivery between sockets over RC or TCP/IP. RDS is intended for use with Oracle RAC 11g.

For programming details, enter:

```
host1$ man rds
```

4.2.2 RDS Configuration

The RDS ULP is installed as part of Mellanox OFED for Linux. To load the RDS module upon boot, edit the file `/etc/infiniband/openib.conf` and set “RDS_LOAD=yes”.

For the changes to take effect, run: **`/etc/init.d/openibd restart`**

4.3 Sockets Direct Protocol

4.3.1 Overview

Sockets Direct Protocol (SDP) is an InfiniBand byte-stream transport protocol that provides TCP stream semantics. Capable of utilizing InfiniBand's advanced protocol offload capabilities, SDP can provide lower latency, higher bandwidth, and lower CPU utilization than IPoIB or Ethernet running some sockets-based applications.

SDP can be used by applications and improve their performance transparently (that is, without any recompilation). Since SDP has the same socket semantics as TCP, an existing application is able to run using SDP; the difference is that the application’s TCP socket gets replaced with an SDP socket.

It is also possible to configure the driver to automatically translate TCP to SDP based on the source IP/port, the destination, or the application name. See Section 4.3.5.

The SDP protocol is composed of a kernel module that implements the SDP as a new address-family/protocol-family, and a library (see Section 4.3.2) that is used for replacing the TCP address family with SDP according to a policy.

This chapter includes the following sections:

- Section 4.3.2,“libsdp.so Library,” on page 46

- Section 4.3.3,“Configuring SDP,” on page 46
- Section 4.3.4,“Environment Variables,” on page 49
- Section 4.3.5,“Converting Socket-based Applications,” on page 49
- Section 4.3.6,“BZCopy – Zero Copy Send,” on page 55
- Section 4.3.7,“Using RDMA for Small Buffers,” on page 55

4.3.2 libsdp.so Library

`libsdp.so` is a dynamically linked library, which is used for transparent integration of applications with SDP. The library is preloaded, and therefore takes precedence over glibc for certain socket calls. Thus, it can transparently replace the TCP socket family with SDP socket calls.

The library also implements a user-level socket switch. Using a configuration file, the system administrator can set up the policy that selects the type of socket to be used. `libsdp.so` also has the option to allow server sockets to listen on both SDP and TCP interfaces. The various configurations with SDP/TCP sockets are explained inside the `/etc/libsdp.conf` file.

4.3.3 Configuring SDP

To load SDP upon boot, edit the file `/etc/infiniband/openib.conf` and set “SDP_LOAD=yes”.

For the changes to take effect, run: **`/etc/init.d/openibd restart`**

SDP can work over IPoIB interfaces or RoCE interfaces. In case of IPoIB, SDP uses the same IP addresses and interface names as IPoIB (see IPoIB configuration in Section 4.6.3 and Section 4.6.3.3). In case of RoCE, SDP use the same IP addresses and interface names of the corresponding mlx4_en interfaces (see mlx4_en configuration in Section 5.4 and Section 5.4.4).

4.3.3.1 How to Know SDP Is Working

Since SDP is a transparent TCP replacement, it can sometimes be difficult to know that it is working correctly. To check whether traffic is passing through SDP or TCP, monitor the file / proc/net/sdpstats and see which counters are running.

Alternative Method – Using the sdpnetstat Program

The `sdpnetstat` program can be used to verify both that SDP is loaded and is being used. The following command shows all active SDP sockets using the same format as the traditional `netstat` program. Without the ‘-S’ option, it shows all the information that `netstat` does plus SDP data.

```
host1$ sdpnetstat -S
```

Assuming that the SDP kernel module is loaded and is being used, then the output of the command will be as follows:

```
host1$ sdpnetstat -S
Proto Recv-Q Send-Q Local Address           Foreign Address
sdp        0      0 193.168.10.144:34216    193.168.10.125:12865
sdp        0 884720 193.168.10.144:42724    193.168.10.:filenet-rmi
```

The example output above shows two active SDP sockets and contains details about the connections.

If the SDP kernel module is not loaded, then the output of the command will be something like the following:

```
host1$ sdpnetstat -S
Proto Recv-Q Send-Q Local Address           Foreign Address
netstat: no support for `AF INET (tcp)' on this system.
```

To verify whether the module is loaded or not, you can use the `lsmod` command:

```
ib_sdp1250200
```

The example output above shows that the SDP module is loaded.

If the SDP module *is* loaded and the `sdpnetstat` command did not show SDP sockets, then SDP is not being used by any application.

4.3.3.2 Monitoring and Troubleshooting Tools

SDP has debug support for both the user space `libsdp.so` library and the `ib_sdp` kernel module.. Both can be useful to understand why a TCP socket was not redirected over SDP and to help find problems in the SDP implementation.

User Space SDP Debug

User-space SDP debug is controlled by options in the `libsdp.conf` file. You can also have a local version and point to it explicitly using the following command:

```
host1$ export LIBSDP_CONFIG_FILE=<path>/libsdp.conf
```

To obtain extensive debug information, you can modify `libsdp.conf` to have the `log` directive produce maximum debug output (provide the `min-level` flag with the value 1).

The **`log`** statement enables the user to specify the debug and error messages that are to be sent and their destination. The syntax of **`log`** is as follows:

```
log [destination (stderr | syslog | file <filename>)] [min-level 1-9]
```

where options are:

```
destination      send log messages to the specified destination:
                 stderr: forward messages to the STDERR
                 syslog: send messages to the syslog service
                 file <filename>: write messages to the file
                 /var/log/filename for root. For a regular
                 user, write to /tmp/<filename>.<uid> if filename is
                 not specified as a full path; otherwise, write to
                 <path>/<filename>.<uid>
min-level        verbosity level of the log:
                 9: print errors only
                 8: print warnings
                  7: print connect and listen summary (useful for tracking
                    SDP usage)
                 4: print positive match summary (useful for config file
                    debug)
                 3: print negative match summary (useful for config file
                    debug)
                 2: print function calls and return values
                 1: print debug messages
```

Examples:

To print SDP usage per connect and listern to STDERR, include the following statement:

```
log min-level 7 destination stderr
```

A non-root user can configure `libsdp.so` to record function calls and return values in the file `/tmp/libsdp.log.<pid>` (root log goes to `/var/log/libsdp.log` for this example) by including the following statement in `libsdp.conf`:

```
log min-level 2 destination file libsdp.log
```

To print errors only to syslog, include the following statement:

```
log min-level 9 destination syslog
```

To print maximum output to the file `/tmp/sdp_debug.log.<pid>`, include the following statement:

```
log min-level 1 destination file sdp_debug.log
```

Kernel Space SDP Debug

The SDP kernel module can log detailed trace information if you enable it using the 'debug_level' variable in the sysfs filesystem. The following command performs this:

```
host1$ echo 1 > /sys/module/ib_sdp/parameters/sdp_debug_level
```

Depending on the operating system distribution on your machine, you may need an extra level—parameters— in the directory structure, so you may need to direct the echo command to /sys/module/ib_sdp/parameters/debug_level.

Turning off kernel debug is done by setting the sysfs variable to zero using the following command:

```
host1$ echo 0 > /sys/module/ib_sdp/parameters/sdp_debug_level
```

To display debug information, use the `dmesg` command:

```
host1$ dmesg
```

4.3.4 Environment Variables

For the transparent integration with SDP, the following two environment variables are required:

1. LD_PRELOAD – this environment variable is used to preload `libsdp.so` and it should point to the `libsdp.so` library. The variable should be set by the system administrator to `/usr/lib/libsdp.so (or /usr/lib64/libsdp.so)`.
2. LIBSDP_CONFIG_FILE – this environment variable is used to configure the policy for replacing TCP sockets with SDP sockets. By default it points to: `/etc/libsdp.conf`.
3. SIMPLE_LIBSDP – ignore `libsdp.conf` and always use SDP

4.3.5 Converting Socket-based Applications

You can convert a socket-based application to use SDP instead of TCP in an automatic (also called transparent) mode or in an explicit (also called non-transparent) mode.

Automatic (Transparent) Conversion

The `libsdp.conf` configuration (policy) file is used to control the automatic transparent replacement of TCP sockets with SDP sockets. In this mode, socket streams are converted based upon a destination port, a listening port, or a program name.

Socket control statements in `libsdp.conf` allow the user to specify when `libsdp` should replace AF_INET/SOCK_STREAM sockets with AF_SDP/SOCK_STREAM sockets. Each control statement specifies a matching rule that applies if all its subexpressions must evaluate as true (logical and).

The `use` statement controls which type of sockets to open. The format of a `use` statement is as follows:

```
use <address-family> <role> <program-name|*> <address|*>:<port range|*>
```

where

```
<address-family>
               can be one of
```

```
                sdp: for specifying when an SDP should be used
                tcp: for specifying when an SDP socket should not be
                matched
                both: for specifying when both SDP and AF_INET sockets
                should be used
                Note that both semantics is different for server and
                 client roles. For server, it means that the server will
                be listening on both SDP and TCP sockets. For client,
                the connect function will first attempt to use SDP and
                will silently fall back to TCP if the SDP connection
                fails.
<role>
                can be one of
                  server or listen: for defining the listening port address
                family
                client or connect: for defining the connected port
                address family
<program-name|*>
                 Defines the program name the rule applies to (not includ
                  ing the path). Wildcards with same semantics as 'ls' are
                 supported (* and ?). So db2* would match on any program
                  with a name starting with db2. t?cp would match on ttcp,
                 etc.
                 If program-name is not provided (default), the statement
                 matches all programs.
<address|*>
                 Either the local address to which the server binds,
                 or the remote server address to which the client
                 connects. The syntax for address matching is:
                 <IPv4 address>[/<prefix_length>]|*
                 IPv4 address = [0-9]+\.[0-9]+\.[0-9]+\.[0-9]+ each sub
                 number < 255
                 prefix_length = [0-9]+ and with value <= 32.
                 A prefix_length of 24  matches the subnet mask
                 255.255.255.0.
                  A prefix_length of 32 requires  matching of the exact IP.
<port range>
                 start-port[-end-port] where port numbers are >0 and <65536
```

Note that rules are evaluated in the order of definition. So the first match wins. If no match is made, `libsdp` will default to `both`.

Examples:

- Use SDP by clients connecting to machines that belongs to subnet 192.168.1.*

```
use sdp connect *  192.168.1.0/24:*
```

- Use SDP by ttcp when it connects to port 5001 of any machine

```
use sdp    listen  ttcp      *:5001
```

- Use TCP for any program with name starting with ttcp* serving ports 22 to 25

```
use tcp    server  ttcp*     *:22-25
```

- Listen on both TCP and SDP by any server that listen on port 8080

```
use both   server  *         *:8080
```

- Connect ssh through SDP and fallback to TCP to hosts on 11.4.8.* port 22

```
use both   connect *         11.4.8.0/24:22
```

Explicit (Non-transparent) Conversion

Use explicit conversion if you need to maintain full control from your application while using SDP. To configure an explicit conversion to use SDP, simply recompile the application replacing PF_INET (or PF_INET) with AF_INET_SDP (or AF_INET_SDP) when calling the `socket()` system call in the source code. The value of AF_INET_SDP is defined in the file `sdp_socket.h` or you can define it inline:

```
#define AF_INET_SDP 27
#define PF_INET_SDP AF_INET_SDP
```

You can compile and execute the following very simple TCP application that has been converted explicitly to SDP:

Compilation:

```
gcc sdp_server.c -o sdp_server
gcc sdp_client.c -o sdp_client
Usage:
Server:
    host1$ sdp_server
Client:
    host1$ sdp_client <server IP addr>
```

Example:

Server:

```
host1$ ./sdp_server
accepted connection from 15.2.2.42:48710
read 2048 bytes
end of test
host1$
```

Client:

```
host2$  ./sdp_client 15.2.2.43
connected to 15.2.2.43:22222
sent 2048 bytes
host2$
sdp_client.c Code

/*
 *  usage: ./sdp_client <ip_addr>
 */

#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <unistd.h>
#include <string.h>

#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

#define DEF_PORT 22222

#define AF_INET_SDP 27
#define PF_INET_SDP AF_INET_SDP

#define TXBUFSZ 2048
uint8_t tx_buffer[TXBUFSZ];

int
main(int argc, char **argv)
{
    if ( argc < 2) {
        printf("Usage: sdp_client <ip_addr>\n");
        exit(EXIT_FAILURE);
    }

    int sd = socket(PF_INET_SDP, SOCK_STREAM, 0);
    if ( sd < 0) {
        perror("socket() failed");
        exit(EXIT_FAILURE);
    }

    struct sockaddr_in to_addr = {
```

```
        .sin_family = AF_INET,
        .sin_port = htons(DEF_PORT),
    };

    int ip_ret = inet_aton(argv[1], &to_addr.sin_addr);
    if ( ip_ret == 0) {
        printf("invalid ip address '%s'\n", argv[1]);
        exit(EXIT_FAILURE);
    }

    int conn_ret = connect(sd, (struct sockaddr *) &to_addr, sizeof(to_addr));
    if ( conn_ret < 0) {
        perror("connect() failed");
        exit(EXIT_FAILURE);
    }

    printf("connected to %s:%u\n",
            inet_ntoa(to_addr.sin_addr),
            ntohs(to_addr.sin_port) );


    ssize_t nw = write(sd, tx_buffer, TXBUFSZ);
    if ( nw < 0) {
        perror("write() failed");
        exit(EXIT_FAILURE);
    } else if ( nw == 0) {
        printf("socket was closed by remote host\n");
    }

    printf("sent %zd bytes\n", nw);

    close(sd);

    return 0;
}
```

sdp_server.c Code

```
/*
 * Usage: ./sdp_server
 */

#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <unistd.h>
```

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <sys/epoll.h>
#include <errno.h>
#include <assert.h>

#define RXBUFSZ 2048

uint8_t rx_buffer[RXBUFSZ];

#define DEF_PORT 22222

#define AF_INET_SDP 27
#define PF_INET_SDP AF_INET_SDP

int
main(int argc, char **argv)
{

    int sd = socket(PF_INET_SDP, SOCK_STREAM, 0);
    if ( sd < 0) {
        perror("socket() failed");
        exit(EXIT_FAILURE);
    }

    struct sockaddr_in my_addr = {
        .sin_family = AF_INET,
        .sin_port = htons(DEF_PORT),
        .sin_addr.s_addr = INADDR_ANY,
    };


    int retbind = bind(sd, (struct sockaddr *) &my_addr, sizeof(my_addr) );
    if ( retbind < 0) {
        perror("bind() failed");
        exit(EXIT_FAILURE);
    }

    int retlisten = listen(sd, 5/*backlog*/);
    if ( retlisten < 0) {
        perror("listen() failed");
        exit(EXIT_FAILURE);
    }
```

```
    // accept the client connection
    struct sockaddr_in client_addr;

    socklen_t client_addr_len = sizeof(client_addr);
    int cd = accept(sd, (struct sockaddr *) &client_addr, &client_addr_len);
    if ( cd < 0) {
        perror("accept() failed");
        exit(EXIT_FAILURE);
    }

    printf("accepted connection from %s:%u\n",
            inet_ntoa(client_addr.sin_addr),
            ntohs(client_addr.sin_port) );

    ssize_t nr = read(cd, rx_buffer, RXBUFSZ);
    if ( nr < 0) {
        perror("read() failed");
        exit(EXIT_FAILURE);
    } else if ( nr == 0) {
        printf("socket was closed by remote host\n");
    }

    printf("read %zd bytes\n", nr);

    printf("end of test\n");

    close(cd);
    close(sd);

    return 0;
}
```

4.3.6 BZCopy – Zero Copy Send

BZCOPY mode is only effective for large block transfers. By setting the `/sys` parameter `'sdp_zcopy_thresh'` to a non-zero value, a non-standard SDP speedup is enabled. Messages longer than `'sdp_zcopy_thresh'` bytes in length cause the user space buffer to be pinned and the data to be sent directly from the original buffer. This results in less CPU usage and, on many systems, much higher bandwidth.

Note that the default value of `'sdp_zcopy_thresh'` is 64KB, but is may be too low for some systems. You will need to experiment with your hardware to find the best value.

4.3.7 Using RDMA for Small Buffers

For smaller buffers, the overhead of preparing a user buffer to be RDMA'ed is too big; therefore, it is more efficient to use BCopy. (Large buffers can also be sent using RDMA, but they lower CPU utilization.) This mode is called “ZCopy combined mode”. The sendmsg syscall is blocked

until the buffer is transfered to the socket's peer, and the data is copied directly from the user buffer at the source side to the user buffer at the sink side.

To set the threshold, use the module parameter sdp_zcopy_thresh. This parameter can be accessed through sysfs (/sys/module/ib_sdp/parameters/sdp_zcopy_thresh). Setting it to 0, disables ZCopy.

4.4 SCSI RDMA Protocol

4.4.1 Overview

As described in Section 1.4.4, the SCSI RDMA Protocol (SRP) is designed to take full advantage of the protocol offload and RDMA features provided by the InfiniBand architecture. SRP allows a large body of SCSI software to be readily used on InfiniBand architecture. The SRP Initiator controls the connection to an SRP Target in order to provide access to remote storage devices across an InfiniBand fabric. The SRP Target resides in an IO unit and provides storage services.

Section 4.4.2 describes the SRP Initiator included in Mellanox OFED for Linux. This package, however, does *not* include an SRP Target.

4.4.2 SRP Initiator

This SRP Initiator is based on open source from OpenFabrics (www.openfabrics.org) that implements the SCSI RDMA Protocol-2 (SRP-2). SRP-2 is described in Document # T10/1524-D available from http://www.t10.org.

The SRP Initiator supports

- Basic SCSI Primary Commands -3 (SPC-3)

  (www.t10.org/ftp/t10/drafts/spc3/spc3r21b.pdf)
- Basic SCSI Block Commands -2 (SBC-2)

  (www.t10.org/ftp/t10/drafts/sbc2/sbc2r16.pdf)
- Basic functionality, task management and limited error handling

4.4.2.1 Loading SRP Initiator

To load the SRP module, either execute the “modprobe ib_srp” command after the OFED driver is up, or change the value of SRP_LOAD in `/etc/infiniband/openib.conf` to “yes”.

For the changes to take effect, run: /etc/init.d/openibd restart

When loading the ib_srp module, it is possible to set the module parameter srp_sg_tablesize. This is the maximum number of gather/scatter entries per I/O (default: 12).

4.4.2.2 Manually Establishing an SRP Connection

The following steps describe how to manually load an SRP connection between the Initiator and an SRP Target. Section 4.4.2.4 explains how to do this automatically.

- Make sure that the `ib_srp` module is loaded, the SRP Initiator is reachable by the SRP Target, and that an SM is running.
- To establish a connection with an SRP Target and create an SRP (SCSI) device for that target under `/dev`, use the following command:

```
echo -n id_ext=[GUID value],ioc_guid=[GUID value],dgid=[port GID value],\
pkey=ffff,service_id=[service[0] value] > \
/sys/class/infiniband_srp/srp-mthca[hca number]-[port number]/add_target
```

  See Section 4.4.2.3 for instructions on how the parameters in this `echo` command may be obtained.

  Notes:

  - Execution of the above "echo" command may take some time
  - The SM must be running while the command executes
  - It is possible to include additional parameters in the echo command:
    - max_cmd_per_lun - Default: 63
    - max_sect (short for max_sectors) - sets the request size of a command
    - io_class - Default: 0x100 as in rev 16A of the specification

      (In rev 10 the default was 0xff00)
    - initiator_ext - Please refer to Section 9 (Multiple Connections...)
- To list the new SCSI devices that have been added by the echo command, you may use either of the following two methods:
  - Execute "fdisk -l". This command lists all devices; the new devices are included in this listing.
  - Execute "dmesg" or `look at /var/log/messages` to find messages with the names of the new devices.

4.4.2.3 SRP Tools - ibsrpdm and srp_daemon

To assist in performing the steps in Section 6, the OFED distribution provides two utilities, ibsrpdm and srp_daemon, which

- Detect targets on the fabric reachable by the Initiator (for Step 1)
- Output target attributes in a format suitable for use in the above "echo" command (Step 2)

The utilities can be found under `/usr/sbin/`, and are part of the srptools RPM that may be installed using the Mellanox OFED installation. Detailed information regarding the various options for these utilities are provided by their man pages.

Below, several usage scenarios for these utilities are presented.

ibsrpdm

ibsrpdm is using for the following tasks:

1. Detecting reachable targets

a. To detect all targets reachable by the SRP initiator via the default umad device (/dev/umad0), execute the following command:

```
ibsrpdm
```

This command will output information on each SRP Target detected, in human-readable form.

Sample output:

```
     IO Unit Info:
          port LID:        0103
          port GID:        fe80000000000000002c90200402bd5
          change ID:       0002
          max controllers: 0x10
       controller[  1]
           GUID:      0002c90200402bd4
           vendor ID: 0002c9
           device ID: 005a44
           IO class : 0100
           ID:        LSI Storage Systems SRP Driver 200400a0b81146a1
           service entries: 1
           service[  0]: 200400a0b81146a1 / SRP.T10:200400A0B81146A1
```

b. To detect all the SRP Targets reachable by the SRP Initiator via another umad device, use the following command:

```
ibsrpdm -d <umad device>
```

2. Assistance in creating an SRP connection

a. To generate output suitable for utilization in the “echo” command of Section 4.4.2.2, add the ‘-c’ option to ibsrpdm:

```
ibsrpdm -c
```

Sample output:

```
id_ext=200400A0B81146A1,ioc_guid=0002c90200402bd4,
dgid=fe80000000000000002c90200402bd5,pkey=ffff,service_id=200400a0b81146a1
```

b. To establish a connection with an SRP Target using the output from the ‘`libsrpdm -c`’ example above, execute the following command:

```
echo -n id_ext=200400A0B81146A1,ioc_guid=0002c90200402bd4,
dgid=fe80000000000000002c90200402bd5,pkey=ffff, service_id=200400a0b81146a1 > /sys/
class/infiniband_srp/srp-mthca0-1/add_target
```

The SRP connection should now be up; the newly created SCSI devices should appear in the listing obtained from the ‘`fdisk -l`’ command.

srp_daemon

The srp_daemon utility is based on ibsrpdm and extends its functionality. In addition to the ibsrpdm functionality described above, srp_daemon can also

- Establish an SRP connection by itself (without the need to issue the “echo” command described in Section 4.4.2.2)
- Continue running in background, detecting new targets and establishing SRP connections with them (daemon mode)
- Discover reachable SRP Targets given an infiniband HCA name and port, rather than just by
  /dev/umad<N> where <N> is a digit
- Enable High Availability operation (together with Device-Mapper Multipath)
- Have a configuration file that determines the targets to connect to

1. `srp_daemon` commands equivalent to ibsrpdm:

```
"srp_daemon -a -o"    is equivalent to "ibsrpdm"
"srp_daemon -c -a -o" is equivalent to "ibsrpdm -c"
```

These srp_daemon commands can behave differently than the equivalent ibsrpdm command when /etc/srp_daemon.conf is not empty.

2. `srp_daemon` extensions to `ibsrpdm`
   - To discover SRP Targets reachable from the HCA device <InfiniBand HCA name> and the port <port num>, (and to generate output suitable for 'echo',) you may execute:

```
host1# srp_daemon -c -a -o -i <InfiniBand HCA name> -p <port number>
```

To obtain the list of InfiniBand HCA device names, you can either use the ibstat tool or run ‘ls /sys/class/infiniband’.

   - To both discover the SRP Targets and establish connections with them, just add the -e option to the above command.
   - Executing srp_daemon over a port without the -a option will only display the reachable targets via the port and to which the initiator is not connected. If executing with the -e option it is better to omit -a.
   - It is recommended to use the -n option. This option adds the initiator_ext to the connecting string. (See Section 4.4.2.5 for more details).
   - srp_daemon has a configuration file that can be set, where the default is /etc/srp_daemon.conf. Use the -f to supply a different configuration file that configures the targets srp_daemon is allowed to connect to. The configuration file can also be used to set values for additional parameters (e.g., max_cmd_per_lun, max_sect).

- A continuous background (daemon) operation, providing an automatic ongoing detection and connection capability. See Section 4.4.2.4.

4.4.2.4 Automatic Discovery and Connection to Targets

- Make sure that the ib_srp module is loaded, the SRP Initiator can reach an SRP Target, and that an SM is running.
- To connect to all the existing Targets in the fabric, run "**`srp_daemon -e -o`**". This utility will scan the fabric once, connect to every Target it detects, and then exit.

srp_daemon will follow the configuration it finds in /etc/srp_daemon.conf. Thus, it will ignore a target that is disallowed in the configuration file.

- To connect to all the existing Targets in the fabric and to connect to new targets that will join the fabric, execute srp_daemon -e. This utility continues to execute until it is either killed by the user or encounters connection errors (such as no SM in the fabric).
- To execute SRP daemon as a daemon you may run "**`run_srp_daemon`**" (found under `/usr/sbin/`), providing it with the same options used for running `srp_daemon`.

Make sure only one instance of run_srp_daemon runs per port.

- To execute SRP daemon as a daemon on all the ports, run "**`srp_daemon.sh`**" (found under `/usr/sbin/`). `srp_daemon.sh` sends its log to `/var/log/srp_daemon.log`.
- It is possible to configure this script to execute automatically when the InfiniBand driver starts by changing the value of SRPHA_ENABLE in `/etc/infiniband/openib.conf` to "yes". However, this option also enables SRP High Availability that has some more features – see Section 4.4.2.6).

  For the changes in `openib.conf` to take effect, run:
  `/etc/init.d/openibd restart`

4.4.2.5 Multiple Connections from Initiator IB Port to the Target

Some system configurations may need multiple SRP connections from the SRP Initiator to the same SRP Target: to the same Target IB port, or to different IB ports on the same Target HCA.

In case of a single Target IB port, i.e., SRP connections use the same path, the configuration is enabled using a different initiator_ext value for each SRP connection. The initiator_ext value is a 16-hexadecimal-digit value specified in the connection command.

Also in case of two physical connections (i.e., network paths) from a single initiator IB port to two different IB ports on the same Target HCA, there is need for a different initiator_ext value on each path. The conventions is to use the Target port GUID as the initiator_ext value for the relevant path.

If you use srp_daemon with -n flag, it automatically assigns initiator_ext values according to this convention. For example:

```
 id_ext=200500A0B81146A1,ioc_guid=0002c90200402bec,\
dgid=fe8000000000000002c90200402bed,pkey=ffff,\
service_id=200500a0b81146a1,initiator_ext=ed2b400002c90200
```

Notes:

1. It is recommended to use the -n flag for all srp_daemon invocations.
2. ibsrpdm does not have a corresponding option.
3. srp_daemon.sh always uses the -n option (whether invoked manually by the user, or automatically at startup by setting SRPHA_ENABLE to yes).

4.4.2.6 High Availability (HA)

Overview

High Availability works using the Device-Mapper (DM) multipath and the SRP daemon. Each initiator is connected to the same target from several ports/HCAs. The DM multipath is responsible for joining together different paths to the same target and for fail-over between paths when one of them goes offline. Multipath will be executed on newly joined SCSI devices.

Each initiator should execute several instances of the SRP daemon, one for each port. At startup, each SRP daemon detects the SRP Targets in the fabric and sends requests to the ib_srp module to connect to each of them. These SRP daemons also detect targets that subsequently join the fabric, and send the ib_srp module requests to connect to them as well.

Operation

When a path (from port1) to a target fails, the ib_srp module starts an error recovery process. If this process gets to the reset_host stage and there is no path to the target from this port, ib_srp will remove this scsi_host. After the scsi_host is removed, multipath switches to another path to this target (from another port/HCA).

When the failed path recovers, it will be detected by the SRP daemon. The SRP daemon will then request ib_srp to connect to this target. Once the connection is up, there will be a new scsi_host for this target. Multipath will be executed on the devices of this host, returning to the original state (prior to the failed path).

Prerequisites

Installation for RHEL4/5: (Execute once)

- Verify that the standard device-mapper-multipath rpm is installed. If not, install it from the RHEL distribution.

Installation for SLES10: (Execute once)

- Verify that multipath is installed. If not, take it from the installation (you may use ‘`yast`’).
- Update udev: (Execute once - for manual activation of High Availability only)
- Add a file to /etc/udev/rules.d/ (you can call it 91-srp.rules). This file should have one line:

```
ACTION=="add", KERNEL=="sd*[!0-9]", RUN+="/sbin/multipath %M:%m"
```

When SRPHA_ENABLE is set to "yes" (see Automatic Activation of High Availability below), this file is created upon each boot of the driver and is deleted when the driver is unloaded.

Manual Activation of High Availability

Initialization: (Execute after each boot of the driver)

1. Execute modprobe dm-multipath
2. Execute modprobe ib-srp
3. Make sure you have created file /etc/udev/rules.d/91-srp.rules as described above.
4. Execute for each port and each HCA:

```
srp_daemon -c -e -R 300 -i <InfiniBand HCA name> -p <port number>
```

This step can be performed by executing srp_daemon.sh, which sends its log to /var/log/srp_daemon.log.

Now it is possible to access the SRP LUNs on `/dev/mapper/`.

It is possible for regular (non-SRP) LUNs to also be present; the SRP LUNs may be identified by their names. You can configure the /etc/multipath.conf file to change multipath behavior.

It is also possible that the SRP LUNs will not appear under /dev/mapper/. This can occur if the SRP LUNs are in the black-list of multipath. Edit the ‘blacklist’ section in /etc/multipath.conf and make sure the SRP LUNs are not black-listed.

Automatic Activation of High Availability

- Set the value of SRPHA_ENABLE in `/etc/infiniband/openib.conf` to "yes".

  For the changes in `openib.conf` to take effect, run:
  `/etc/init.d/openibd restart`

- From the next loading of the driver it will be possible to access the SRP LUNs on /dev/mapper/

It is possible that regular (not SRP) LUNs may also be present; the SRP LUNs may be identified by their name.

- It is possible to see the output of the SRP daemon in /var/log/srp_daemon.log

4.4.2.7 Shutting Down SRP

SRP can be shutdown by using “rmmod ib_srp”, or by stopping the OFED driver (“`/etc/init.d/openibd stop`”), or as a by-product of a complete system shutdown.

Prior to shutting down SRP, remove all references to it. The actions you need to take depend on the way SRP was loaded. There are three cases:

1. Without High Availability

   When working without High Availability, you should unmount the SRP partitions that were mounted prior to shutting down SRP.

2. After Manual Activation of High Availability

If you manually activated SRP High Availability, perform the following steps:

   a. Unmount all SRP partitions that were mounted.

   b. Kill the SRP daemon instances.

   c. Make sure there are no multipath instances running. If there are multiple instances, wait for them to end or kill them.

   d. Run: `multipath -F`

3. After Automatic Activation of High Availability

   If SRP High Availability was automatically activated, SRP shutdown must be part of the driver shutdown ("/etc/init.d/openibd stop") which performs Steps 2-4 of case b above. However, you still have to unmount all SRP partitions that were mounted before driver shutdown.

4.5 Ethernet over IB (EoIB) vNic

The Ethernet over IB (EoIB) mlx4_vnic module is a network interface implementation over InfiniBand. EoIB encapsulates Layer 2 datagrams over an InfiniBand Datagram (UD) transport service. The InfiniBand UD datagrams encapsulates the entire Ethernet L2 datagram and its payload.

To perform this operation the module performs an address translation from Ethernet layer 2 MAC addresses (48 bits long) to InfiniBand layer 2 addresses made of LID/GID and QPN. This translation is totally invisible to the OS and user. Thus, differentiating EoIB from IPoIB which exposes a 20 Bytes HW address to the OS. The mlx4_vnic module is designed for Mellanox's ConnectX® family of HCAs and intended to be used with Mellanox's BridgeX® gateway family. Having a BridgeX gateway is a requirement for using EoIB. It performs the following operations:

- Enables the layer 2 address translation required by the mlx4_vnic module.
- Enables routing of packets from the InfiniBand fabric to a 1 or 10 GigE Ethernet subnet.

4.5.1 Ethernet over IB Topology

EoIB is designed to work over an InfiniBand fabric and requires the presence of two entities:

- Subnet Manager (SM)

  The required subnet manager configuration is not unique to EoIB but rather similar to other InfiniBand applications and ULPs.

- BridgeX gateway

  The BridgeX gateway is at the heart of EoIB. On one side, usually referred to as the "internal" side, it is connected to the InfiniBand fabric by one or more links. On the other side, usually referred to as the "external" side, it is connected to the Ethernet subnet by one or more ports. The Ethernet connections on the BridgeX's external side are called external ports or eports. Every BridgeX that is in use with EoIB needs to have one or more eports connected.

4.5.1.1 External Ports (eports) and Gateway

The combination of a specific BridgeX box and a specific eport is referred to as a gateway. The gateway is an entity that is visible to the EoIB host driver and is used in the configuration of the network interfaces on the host side. For example, in the host administered vNics the user will request to open an interface on a specific gateway identifying it by the BridgeX box and eport name.

Distinguishing between gateways is essential because they determine the network topology and affect the path that a packet traverses between hosts. A packet that is sent from the host on a specific EoIB interface will be routed to the Ethernet subnet through a specific external port connection on the BridgeX box.

4.5.1.2 Virtual Hubs (vHubs)

Virtual hubs connect zero or more EoIB interfaces (on internal hosts) and an eport through a virtual hub. Each vHub has a unique virtual LAN (VLAN) ID. Virtual hub participants can send packets to one another directly without the assistance of the Ethernet subnet (external side) routing. This means that two EoIB interfaces on the same vHub will communicate solely using the InfiniBand fabric. EoIB interfaces residing on two different vHubs (whether on the same gateway or not) cannot communicate directly.

There are two types of vHubs:

- a default vHub (one per gateway) without a VLAN ID
- vHubs with unique different VLAN IDs

Each vHub belongs to a specific gateway (BridgeX® + eport), and each gateway has one default vHub, and zero or more VLAN-associated vHubs. A specific gateway can have multiple vHubs distinguishable by their unique VLAN ID. Traffic coming from the Ethernet side on a specific eport will be routed to the relevant vHub group based on its VLAN tag (or to the default vHub for that GW if no vLan ID is present).

4.5.1.3 Virtual NIC (vNic)

A virtual NIC is a network interface instance on the host side which belongs to a single vHub on a specific GW. The vNic behaves similar to any regular hardware network interface. The host can have multiple interfaces that belong to the same vHub.

4.5.2 EoIB Configuration

mlx4_vnic module supports two different modes of configuration which is passed to the host mlx4_vnic driver using the EoIB protocol:

- host administration where the vNic is configured on the host side
- network administration where the configuration is done by the BridgeX

Both modes of operation require the presence of a BridgeX gateway in order to work properly. The EoIB driver supports a mixture of host and network administered vNics.

4.5.2.1 EoIB Host Administered vNic

In the host administered mode, vNics are configured using static configuration files located on the host side. These configuration files define the number of vNics, and the vHub that each host administered vNic will belong to (i.e., the vNic's BridgeX box, eport and VLAN id properties). The mlx4_vnic_confd service is used to read these configuration files and pass the relevant data to the mlx4_vnic module. EoIB Host Administered vNic supports two forms of configuration files:

- “Central Configuration File - /etc/infiniband/mlx4_vnic.conf”
- “vNic Specific Configuration Files - ifcfg-ethX”

Both forms of configuration supply the same functionality. If both forms of configuration files exist, the central configuration file has precedence and only this file will be used.

Central Configuration File - /etc/infiniband/mlx4_vnic.conf

The mlx4_vnic.conf file consists of lines, each describing one vNic. The following file format is used:

```
name=eth47 mac=00:25:8B:27:16:84 ib_port=mlx4_0:1 vid=2 vnic_id=7 bx=BX001 eport=A11
```

The fields used in the file have the following meaning:

Table 2 - mlx4_vnic.conf file format

Field	Description
name	The name of the interface that is displayed when running ifconfig.
mac	The mac address to assign to the vNic.
ib_port	The device name and port number in the form [device name]:[port number]. The device name can be retrieved by running ibv_devinfo and using the output of hca_id field. The port number can have a value of 1 or 2.
vid	[Optional field] If VLAN ID exists, the vNic will be assigned the specified VLAN ID. This value must be between 0 and 4095. • If the vid is set to 'all', the ALL-VLAN mode will be enabled and the vNic will support multiple vNic tags. • If no vid is specified or value -1 is set, the vNic will be assigned to the default vHub associated with the GW.
vnic_id	A unique number per vNic between 0 and 16K.
bx	The BridgeX box system GUID or system name string.

Table 2 - mlx4_vnic.conf file format

Field	Description
eport	The string describing the eport name.
pkey	[Optional field] If discovery_pkey module parameter is set, this value will control which partitions would be used to discover the gateways. For more information about discovery_pkeys please refer to Section 4.5.3.6, “Discovery Partitions Configuration,” on page 72

vNic Specific Configuration Files - ifcfg-ethX

EoIB configuration can use the ifcfg-ethX files used by the network service to derive the needed configuration. In such case, a separate file is required per vNic. Additionally, you need to update the ifcfg-ethX file and add some new attributes to it.

On Red Hat Linux, the new file will be of the form:

```
DEVICE=eth2
HWADDR=00:30:48:7d:de:e4
BOOTPROTO=dhcp
ONBOOT=yes
BXADDR=BX001
BXEPORT=A10
VNICIBPORT=mlx4_0:1
VNICVLAN=3  (Optional field)
GW_PKEY=0xfff1
```

The fields used in the file for vNic configuration have the following meaning:

Table 3 - Red Hat Linux mlx4_vnic.conf file format

Field	Description
DEVICE	An optional field. The name of the interface that is displayed when running ifconfig. If it is not present, the trailer of the configuration file name (e.g. ifcfg-eth47 => "eth47") is used instead.
HWADDR	The mac address to assign the vNic.
BXADDR	The BridgeX box system GUID or system name string.
BXEPORT	The string describing the eport name.
VNICVLAN	[Optional field] If it exists, the vNic will be assigned the VLAN ID specified. This value must be between 0 and 4095 or 'all' for ALL-VLAN feature.
VNICIBPORT	The device name and port number in the form [device name]:[port number]. The device name can be retrieved by running ibv_devinfo and using the output of hca_id field. The port number can have a value of 1 or 2.
GW_PKEY	[Optional field] If discovery_pkey module parameter is set, this value will control on what partition would be used to discover the gateways. For more information about discovery_pkeys please refer to Section 4.5.3.6, “Discovery Partitions Configuration,” on page 72

Other fields available for regular eth interfaces in the ifcfg-ethX files may also be used.

mlx4_vnic_confd

Once the configuration files are updated, the host administered vNics can be created. To manage the host administrated vNics, run the following script:

```
Usage: /etc/init.d/mlx4_vnic_confd {start|stop|restart|reload|status}
```

This script manages host administrated vNics only, to retrieve general information about the vNics on the system including network administrated vNics, refer to Section 4.5.3.1, “mlx4_vnic_info,” on page 69.

When using BXADDR/bx field, all vNics BX address configuration should be consistent: either all of them use GUID format, or name format.

The MAC and VLAN values are set using the configuration files only, other tools such as (vconfig) for VLAN modification, or (ifconfig) for MAC modification, are not supported.

4.5.2.2 EoIB Network Administered vNic

In network administered mode, the configuration of the vNic is done by the BridgeX®. If a vNic is configured for a specific host, it will appear on that host once a connection is established between the BridgeX and the mlx4_vnic module. This connection between the mlx4_vnic modules and all available BridgeX boxes is established automatically when the mlx4_vnic module is loaded. If the BridgeX is configured to remove the vNic, or if the connection between the host and BridgeX is lost, the vNic interface will disappear (running ifconfig will not display the interface). Similar to host administered vNics, a network administered, vNic resides on a specific vHub.

For further information on how to configure a network administered vNic, please refer to BridgeX documentation.

To disable network administered vNics on the host side load mlx4_vnic module with the net_admin module parameter set to 0.

4.5.2.3 VLAN Configuration

A vNic instance is associated with a specific vHub group. This vHub group is connected to a BridgeX external port and has a VLAN tag attribute. When creating/configuring a vNic you define the VLAN tag it will use via the vid or the VNICVLAN fields (if these fields are absent, the vNic will not have a VLAN tag). The vNic's VLAN tag will be present in all EoIB packets sent by the vNics and will be verified on all packets received on the vNic. When passed from the InfiniBand to Ethernet,the EoIB encapsulation will be disassembled but the VLAN tag will remain.

For example, if the vNic "eth23" is associated with a vHub that uses BridgeX "bridge01", eport "A10" and VLAN tag 8, all incoming and outgoing traffic on eth23 will use a VLAN tag of 8.

This will be enforced by both BridgeX and destination hosts. When a packet is passed from the internal fabric to the Ethernet subnet through the BridgeX it will have a "true" Ethernet VLAN tag of 8.

The VLAN implementation used by EoIB uses operating systems unaware of VLANs. This is in many ways similar to switch tagging in which an external Ethernet switch adds/strips tags on traffic preventing the need of OS intervention. EoIB does not support OS aware VLANs in the form of vconfig.

Configuring VLANs

To configure VLAN tag for a vNic, add the VLAN tag property to the configuration file in host administrated mode, or configure the vNic on the appropriate vHub in network administered mode. In the host administered mode when a vHub with the requested VLAN tag is not available, the vNIC's login request will be rejected.

- Host administered VLAN configuration in centralized configuration file can be modified as follow:

  Add "vid=<VLAN tag>" or remove vid property for no VLAN

- Host administered VLAN configuration with ifcfg-ethX configuration files can be modified as follow:

  Add "VNICVLAN=<VLAN tag>" or remove VNICVLAN property for no VLAN

Using a VLAN tag value of 0 is not recommended because the traffic using it would not be separated from non VLAN traffic.

For Host administered vNics, VLAN entry must be set in the BridgeX first. For further information, please refer to BridgeX® documentation.

4.5.2.4 EoIB Multicast Configuration

Configuring Multicast for EoIB interfaces is identical to multicast configuration for native Ethernet interfaces.

EoIB maps Ethernet multicast addresses to InfiniBand MGIDs (Multicast GID). It ensures that different vHubs use mutually exclusive MGIDs. Thus preventing vNics on different vHubs from communicating with one another.

4.5.2.5 EoIB and Quality of Service

EoIB enables the use of InfiniBand service levels. The configuration of the SL is performed through the BridgeX and lets you set different data/control service level values per BridgeX® box.

For further information on the use of non default service levels, please refer to BridgeX documentation.

4.5.2.6 IP Configuration Based on DHCP

Setting an EoIB interface configuration based on DHCP (v3.1.2 which is available via www.isc.org) is performed similarly to the configuration of Ethernet interfaces. When setting the EoIB configuration files, verify that it includes following lines:

- For RedHat: BOOTPROTO=dhcp
- For SLES: BOOTPROTO='dchp'

If EoIB configuration files are included, ifcfg-eth<n> files will be installed under /etc/sysconfig/network-scripts/ on a RedHat machine and under /etc/sysconfig/network/ on a SuSE machine.

DHCP Server

Using a DHCP server with EoIB does not require special configuration. The DHCP server can run on a server located on the Ethernet side (using any Ethernet hardware) or on a server located on the InfiniBand side and running EoIB module.

4.5.2.7 Static EoIB Configuration

To configure a static EoIB you can use an EoIB configuration that is not based on DHCP. Static configuration is similar to a typical Ethernet device configuration. For further information on how to configure IP addresses, please refer to your Linux distribution documentation.

Ethernet configuration files are located at /etc/sysconfig/network-scripts/ on a RedHat machine and at /etc/sysconfig/network/ on a SuSE machine.

4.5.2.8 Sub Interfaces (VLAN)

EoIB interfaces do not support creating sub interfaces via the vconfig command, unless working in ALL VLAN mode.. To create interfaces with VLAN, refer to Section , “Configuring VLANs,” on page 68.

4.5.3 Retrieving EoIB Information

4.5.3.1 mlx4_vnic_info

To retrieve information regarding EoIB interfaces, use the script mlx4_vnic_info. This script provides detailed information about a specific vNic or all EoIB vNic interfaces, such as: BX info, IOA info, SL, PKEY, Link state and interface features. If network administered vNics are enabled, this script can also be used to discover the available BridgeX® boxes from the host side.

- To discover the available gateway, run:

```
mlx4_vnic_info -g
```

- To receive the full vNic information of eth10, run:

```
mlx4_vnic_info -i eth10
```

- To receive a shorter information report on eth10, run:

```
mlx4_vnic_info -s eth10
```

- To get help and usage information, run:

```
mlx4_vnic_info --help
```

4.5.3.2 ethtool

ethtool application is another method to retrieve interface information and change its configuration. EoIB interfaces support ethtool similarly to hardware Ethernet interfaces.

The supported Ethtool options include the following options:

```
-c, -C   - Show and update interrupt coalesce options
-g       - Query RX/TX ring parameters
-k, -K   - Show and update protocol offloads
-i       - Show driver information
-S       - Show adapter statistics
```

For more information on ethtool run: ethtool -h

4.5.3.3 Link State

An EoIB interface can report two different link states:

- The physical link state of the interface that is made up of the actual HCA port link state and the status of the vNics connection with the BridgeX®. If the HCA port link state is down or the EoIB connection with the BridgeX has failed, the link will be reported as down because without the connection to the BridgeX the EoIB protocol cannot work and no data can be sent on the wire. The mlx4_vnic driver can also report the status of the external BridgeX port status by using the mlx4_vnic_info script. If the eport_state_enforce module parameter is set, then the external port state will be reported as the vNic interface link state. If the connection between the vNic and the BridgeX is broken (hence the external port state is unknown)the link will be reported as down.
- the link state of the external port associated with the vNic interface

A link state is down on a host administrated vNic, when the BridgeX is connected and the InfiniBand fabric appears to be functional. The issue might result from a misconfiguration of either BXADDR or/and BXEPORT configuration file.

To query the link state run the following command and look for "Link detected":

```
ethtool <interface name>
```

Example:

```
ethtool eth10
Settings for eth10:
Supported ports: [ ]
Supported link modes:
Supports auto-negotiation: No
Advertised link modes:  Not reported
Advertised auto-negotiation: No
Speed: Unknown! (10000)
Duplex: Full
Port: Twisted Pair
PHYAD: 0
Transceiver: internal
Auto-negotiation: off
Supports Wake-on: d
Wake-on: d
Current message level: 0x00000000 (0)
Link detected: yes
```

4.5.3.4 Bonding Driver

EoIB uses the standard Linux bonding driver. For more information on the Linux Bonding driver please refer to:

<kernel-source>/Documentation/networking/bonding.txt.

Currently only fail-over modes are supported by the EoIB driver, load-balancing modes including static and dynamic (LACP) configurations are not supported.

4.5.3.5 Jumbo Frames

EoIB supports jumbo frames up to the InfiniBand limit of 4K bytes. The default Maximum Transmit Unit (MTU) for EoIB driver is 1500 bytes.

To configure EoIB to work with jumbo frames:

1. Make sure that the IB HCA and Switches hardware support 4K MTU.
2. Configure Mellanox low level driver to support 4K MTU. Add:

```
mlx4_core module parameter to set_4k_mtu=1
```

3. Change the MTU value of the vNic, for example, run:

```
 ifconfig eth2 mtu 4038
```

Due to EoIB protocol overhead, the maximum MTU value that can be set for the vNic interface is: 4038 bytes. If the vNic is configured to use VLANs, then the maximum MTU is: 4034 bytes (due to VLAN header insertion).

4.5.3.6 Discovery Partitions Configuration

EoIB enables mapping of VLANs to InfiniBand partitions. Mapping VLANs to partitions causes all EoIB data traffic and all vNic related control traffic to be sent to the mapped partitions. In rare cases, it might be useful to ensure that EoIB discovery packets (packets used for discovery of Gateways (GWs) and vice versa) are sent to a non default partition. This might be used to limit and enforce the visibility of GWs by different hosts.

The **`discovery_pkeys`** module parameter can be used to define which partitions would be used to discove the GWs. The module parameters allow the using of up to 24 different PKEYs. If not set, the default PKEY will be used, and only GWs using the default PKEY would be discovered.

For example, to configure a host to discover GWs on three partitions 0xffff,0xfff1 and 0x3 add the following line to modprobe configuration file:

```
options mlx4_vnic discovery_pkeys=0xffff,0xfff1,0x3
```

When using this feature combined with host administrated vnics, each vnic should also be configured with the partition it should be created on.

For example, for creating host admin vnic on I/F eth20, with pkey 0xfff1 add the following line to ifcg-eth20:

```
GW_PKEY=0xfff1
```

When using a non default partition, the GW partitions should also be configured on the GW in the BridgeX. Additionally, the Subnet Manager must be configured accordingly.

4.5.3.7 ALL VLAN

In Ethernet over InfiniBand (EoIB), a vNic is a member of a vHUB that uniquely defines its Virtual Local Area Networks (VLAN) tag. The VLAN tag is used in the VLAN header within the EoIB packets, and is enforced by EoIB hosts when handling the EoIB packets. The tag is also extended to the Ethernet fabric when packets pass through the BridgeX®. This model of operation ensures a high level of security however, it requires each VLAN tag used to have its own individual vNic to be created and each vHub requires InfiniBand fabric resources like multicast groups (MGIDs).

If many VLANs are needed, the resources required to create and manage them are large. ALL VLAN vHub enables the user to use its resources efficiently by creating a vNic that can support multiple VLAN tags without creating multiple vNics. However, it reduces VLAN separation compared to the vNic /vHub model.

ALL VLAN Functionality

When ALL VLAN is enabled, the address lookup on the BridgeX® consists of the MAC address only (without the VLAN), so all packets with the same MAC regardless of the VLAN, are sent to the same InfiniBand address. Same behavior can be expected from the host EoIB driver, which also sends packets to the relevant InfiniBand addresses while disregarding the VLAN. In both scenarious, the Ethernet packet that is embedded in the EoIB packet includes the VLAN header enabling VLAN enforcement either in the Ethernet fabric or at the receiving EoIB host.

ALL VLAN must be supported by both the BridgeX® and by the host side.

When enabling ALL VLAN, all gateways (LAG or legacy) that have eports belonging to a gateway group (GWG) must be configured to the same behavior.

For example it is impossible to have gateway A2 configured to all-vlan mode and A3 to regular mode, because both belong to GWG A.

A gateway that is configured to work in ALL VLAN mode cannot accept login requests from

- vNics that do not support this mode
- host admin vNics that were not configured to work in ALL VLAN mode, by setting the vlan-id value to a 'all' as as described in Section , “Creating vNICs that Support ALL VLAN Mode,” on page 73.

Creating vNICs that Support ALL VLAN Mode

VLANs are created on a vNIC that supports ALL VLAN mode using "vconfig".

- **net-admin vNics**

  The net-admin vNic supports ALL VLAN mode once it is created on a gateway configured with All-VLAN mode.

- **host-admin vNics**

  - To create an ALL VLAN vnic, set the VLAN's ID to 'all'.

    A gateway that is configured to work in ALL VLAN mode, can only accept login requests from hosts that are also working in a VLAN mode. e.g. the VLAN ID must be set to 'all'.

  This is an example of how to create an ALL VLAN vNic using the mlx4_vnic.conf file:

```
 name=eth44 mac=00:25:8B:27:14:78 ib_port=mlx4_0:1 vid=all vnic_id=5
bx=00:00:00:00:00:00:04:B2 eport=A10
```

  - To create an All-VLAN vNic using a specific configuration file, add the following line to the configuration file:

```
VNICVLAN=all
```

For further information on how to create host-admin vNics, please see Section 4.5.2.1, “EoIB Host Administered vNic,” on page 65 EoIB Host Administered vNic.

Checking the Configuration

To verify the gateway / vNic is configured with the All-VLAN mode, use the mlx4_vnic_info script.

- **Gateway Support**

  To verify the gateway is configured to All-VLAN mode. Run:

  ```
  mlx4_vnic_info -g <GW-NAME>
  ```

  Example:

  ```
  # mlx4_vnic_info -g A2
  IOA_PORT      mlx4_0:1
  BX_NAME       bridge-119c64
  BX_GUID       00:02:c9:03:00:11:61:67
  EPORT_NAME    A2
  EPORT_ID      63
  STATE         connected
  GW_TYPE       LEGACY
  PKEY          0xffff
  ALL_VLAN      yes
  ```

- **vNic Support**

  To verify the vNIC is configured to All-VLAN mode. Run:

  ```
  mlx4_vnic_info -i <interface>
  ```

  Example:

  ```
  # mlx4_vnic_info -i eth204
  NETDEV_NAME   eth204
  NETDEV_LINK   up
  NETDEV_OPEN   yes
  GW_TYPE       LEGACY
  ALL_VLAN      yes
  ```

For further information on mlx4_vnic_info script, please see Section 4.5.3.1, “mlx4_vnic_info,” on page 69.

4.5.4 Advanced EoIB Settings

4.5.4.1 Module Parameters

The mlx4_vnic driver supports the following module parameters. These parameters are intended to enable more specific configuration of the mlx4_vnic driver to customer needs. The mlx4_vnic is also effected by module parameters of other modules such as set_4k_mtu of mlx4_core. This modules are not addressed in this section.

The available module parameters include:

- tx_rings_num: Number of TX rings, use 0 for #cpus [default 0, max 32]
- tx_rings_len: Length of TX rings, must be power of two [default 1024, max 8K]
- rx_rings_num: Number of RX rings, use 0 for #cpus [default 0, max 32]
- rx_rings_len: Length of RX rings, must be power of two [default 2048, max 8K]
- vnic_net_admin: Network administration enabled [default 1]
- lro_num: Number of LRO sessions per ring, use 0 to disable LRO [default 32, max 32]

- eport_state_enforce: Bring vNic up only when corresponding External Port is up [default 0]
- discovery_pkeys: Vector of up to 24 PKEYs to be used for discovery [default 0xFFFF] (array of int)

For all module parameters list and description, run:

```
mlx4_vnic_info -I
```

To check the current module parameters, run:

```
mlx4_vnic_info -P
```

Default RX/TX rings number is the number of logical CPUs (threads). To set non-default values to module parameters, the following line should be added to modprobe configuration file (e.g. /etc/modprobe.conf file):

```
options mlx4_vnic <param_name>=<value> <param_name>=<value> ...
```

For additional information about **discovery_pkeys** please refer to Section 4.5.3.6, “Discovery Partitions Configuration,” on page 72

4.5.4.2 vNic Interface Naming

The mlx4_vnic driver enables the kernel to determine the name of the registered vNic. By default, the Linux kernel assigns each vNic interface the name eth<N>, where <N> is an incremental number that keeps the interface name unique in the system. The vNic interface name may not remain consistent among hosts or BridgeX reboots as the vNic creation can happen in a different order each time. Therefore, the interface name may change because of a "first-come-first-served" kernel policy. In automatic network administered mode, the vNic MAC address may also change, which makes it difficult to keep the interface configuration persistent.

To control the interface name, you can use standard Linux utilities such as IFRENAME(8), IP(8) or UDEV(7). For example, to change the interface eth2 name to eth.bx01.a10, run:

```
ifrename -i eth2 -n eth.bx01.a10
```

To generate a unique vNic interface name, use the mlx4_vnic_info script with the '-u' flag. The script will generate a new name based on the scheme:

```
eth<pci-id>.<ib-port-num>.<gw_port_id>.[vlan-id]
```

For example, if vNic eth2 resides on an InfiniBand card on the PCI BUS ID 0a:00.0 PORT #1, and is connected to the GW PORT ID #3 without VLAN, its unique name will be:

```
mlx4_vnic_info -u eth2
eth2   eth10.1.3
```

You can add your own custom udev rule to use the output of the script and to rename the vNic interfaces automatically. To create a new udev rule file under /etc/udev/rules.d/61-vnic-net.rules, include the line:

```
SUBSYSTEM=="net", PROGRAM=="/sbin/mlx4_vnic_info -u %k", NAME="%c{2+}"
```

UDEV service is active by default however if it is not active, run:

```
/sbin/udevd -d
```

When vNic MAC address is consistent, you can statically name each interface using the UDEV following rule:

```
SUBSYSTEM=="net", SYSFS{address}=="aa:bb:cc:dd:ee:ff", NAME="ethX"
```

For further information on the UDEV rules syntax, please refer to udev man pages.

4.5.4.3 Para-Virtualized vNic

EoIB driver interfaces can be also used for Linux based virtualization environment such as Xen/ KVM based Hypervisors. This section explains how to configure Para-Virtualized (PV) EoIB to work in such an environment.

Driver Configuration

For PV-EoIB to work properly, the following features must be disabled in the driver:

- Large Receive Offload (LRO)
- TX completion polling
- RX fragmented buffers

To disable the features above, edit the modprobe configuration file as follow:

```
options mlx4_vnic lro_num=0 tx_polling=0 rx_linear=1
```

For the full list of mlx4_vnic module parameters, run:

```
# modinfo mlx4_vnic
```

Network Configuration

PV-EoIB supports both L2 (bridged) and L3 (routed) network models. The 'physical' interfaces that can be enslaved to the Hypervisor virtual bridge are actually EoIB vNics, and they can be created as on an native Linux machine. PV-EoIB driver supports both host-administrated and network-administrated vNics. Please refer to Section 4.5.2, “EoIB Configuration,” on page 65 for more information on vNics configuration.

Once an EoIB vNic is enslaved to a virtual bridge, it can be used by any Guest OS that is supported by the Hypervisor. The driver will automatically manage the resources required to serve the Guest OS network virtual interfaces (based on their MAC address).

To see the list of MAC addresses served by an EoIB vNic, log into the Hypervisor and run the command:

```
# mlx4_vnic_info -m <interface>
```

The driver detects virtual interfaces MAC addresses based in their outgoing packets, so you may notice that the virtual MAC address is being detected by the EoIB driver only after the first packet is sent out by the Guest OS. Virtual resources MAC addresses cleanup is managed by mlx4_vnic daemon as explained in Section , “Resources Cleanup,” on page 77.

Multicast Configuration

Virtual machines multicast traffic over PV-EoIB is supported in promiscuous mode. Hence, all multicast traffic is sent over the broadcast domain, and filtered in the VM level.

- To enable promiscuous multicast, log into the BridgeX CLI and run the command:

```
# bxm eoib mcast promiscuous
```

Please refer to BridgeX CLI Guide for additional details.

- To see the multicast configuration of a vNic from the host, log into the Hypervisor and run:

```
# mlx4_vnic_info -i <interface> | grep MCAST
```

VLANs

Virtual LANs are supported in EoIB vNic level, where VLAN tagging/untagging is done by the EoIB driver.

- To enable VLANs on top of a EoIB vNic:
  a. Create a new vNic interface with the corresponding VLAN ID
  b. Enslave it to a virtual bridge to be used by the Guest OS. The VLAN tagging/untagging is transparent to the Guest and managed in EoIB driver level.

The vconfig utility is not supported by EoIB driver, a new vNic instance must be created instead. For further information, see Section 4.5.2.3, “VLAN Configuration,” on page 67.

Virtual Guest Tagging (VGT) is not supported. The model explained above applies to Virtual Switch Tagging (VST) only.

Migration

Some Hypervisors provide the ability to migrate a virtual machine from one physical server to another, this feature is seamlessly supported by PV-EoIB. Any network connectivity over EoIB will automatically be resumed on the new physical server. The downtime that may occur during this process is minor.

Resources Cleanup

When a virtual interface within the Guest OS is no longer connected to an EoIB link, its MAC address need to be removed from the EoIB driver. The cleaning is managed by the Garbage Collector (GC) service. The GC functionality is included in the mlx4_vnic daemon (python script):

```
# /sbin/mlx4_vnicd
```

- To enable/disable the mlx4_vnic daemon,
  a. Edit the **/etc/infiniband/mlx4_vnic.conf** file by including the line:

```
# mlx4_vnicd=<yes|no> [parameters]
```

b. Start the service **mlx4_vnic_confd** to read and apply the configuration:

```
# /etc/init.d/mlx4_vnic_confd start
```

- To see full list of the daemon parameters, run:

```
# mlx4_vnicd --help
```

For example, to enable mlx4_vnic daemon with GC:

```
# cat /etc/infiniband/mlx4_vnic.conf
  mlx4_vnicd=yes gc_enable=yes

# /etc/init.d/mlx4_vnic_confd start
  Checking configuration file:                          [  OK  ]
  Starting mlx4_vnicd (pid 30920):                      [  OK  ]
```

The mlx4_vnicd daemon requires xenstore or libvirt to run.

Some Hypervisors may not have enough memory for the driver domain, as a result **mlx4_vnic** driver may fail to initialize or create more vNics, causing the machine to be unresponsive.

- To avoid this behavior, you can:

a. Allocate more memory for the driver domain.

For further information on how to increase dom0_mem, please refer to:

http://support.citrix.com/article/CTX126531

b. Lower the **mlx4_vnic** driver memory consumption by decreasing its RX/TX rings number and length,

For further information, please refer to Section 4.5.4.1, “Module Parameters,” on page 74.

4.6 IP over InfiniBand

4.6.1 Introduction

The IP over IB (IPoIB) driver is a network interface implementation over InfiniBand. IPoIB encapsulates IP datagrams over an InfiniBand Connected or Datagram transport service. The IPoIB driver, ib_ipoib, exploits the following ConnectX family adapters capabilities:

- Uses any CX IB ports (one or two)
- Inserts IP/UDP/TCP checksum on outgoing packets
- Calculates checksum on received packets
- Support net device TSO through CX LSO capability to defragment large datagrams to MTU quantas.
- Dual operation mode - datagram and connected
- Large MTU support through connected mode

IPoIB also supports the following software based enhancements:

- Large Receive Offload

- NAPI
- Ethtool support

This chapter describes the following:

- IPoIB mode setting (Section 4.6.2)
- IPoIB configuration (Section 4.6.3)
- How to create and remove subinterfaces (Section 4.6.4)
- How to verify IPoIB functionality (Section 4.6.5)
- The ib-bonding driver (Section 4.6.6)

4.6.2 IPoIB Mode Setting

IPoIB can run in two modes of operation: Connected mode and Datagram mode. By default, IPoIB is set to work in Connected mode. This can be changed to become Datagram mode by editing the file `/etc/infiniband/openib.conf` and setting 'SET_IPOIB_CM=no'.

After changing the mode, you need to restart the driver by running:

```
/etc/init.d/openibd restart
```

To check the current mode used for out-going connections, enter:

```
cat /sys/class/net/ib<n>/mode
```

4.6.3 IPoIB Configuration

Unless you have run the installation script `mlnxofedinstall` with the flag '-n', then IPoIB has not been configured by the installation. The configuration of IPoIB requires assigning an IP address and a subnet mask to each HCA port, like any other network adapter card (i.e., you need to prepare a file called ifcfg-ib<n> for each port). The first port on the first HCA in the host is called interface ib0, the second port is called ib1, and so on.

An IPoIB configuration can be based on DHCP (Section 4.6.3.1) or on a static configuration (Section 4.6.3.2) that you need to supply. You can also apply a manual configuration that persists only until the next reboot or driver restart (Section 4.6.3.3).

4.6.3.1 IPoIB Configuration Based on DHCP

Setting an IPoIB interface configuration based on DHCP is performed similarly to the configuration of Ethernet interfaces. In other words, you need to make sure that IPoIB configuration files include the following line:

For RedHat:

```
BOOTPROTO=dhcp
```

For SLES:

```
BOOTPROTO='dchp'
```

If IPoIB configuration files are included, ifcfg-ib<n> files will be installed under:
/etc/sysconfig/network-scripts/ on a RedHat machine
/etc/sysconfig/network/ on a SuSE machine.

A patch for DHCP is required for supporting IPoIB. For further information, please see the REAME which is available under the docs/dhcp/ directory.

Standard DHCP fields holding MAC addresses are not large enough to contain an IPoIB hardware address. To overcome this problem, DHCP over InfiniBand messages convey a client identifier field used to identify the DHCP session. This client identifier field can be used to associate an IP address with a client identifier value, such that the DHCP server will grant the same IP address to any client that conveys this client identifier.

The length of the client identifier field is not fixed in the specification. For the *Mellanox OFED for Linux* package, it is recommended to have IPoIB use the same format that FlexBoot uses for this client identifier – see Section A.3.2, “Configuring the DHCP Server,” on page 195.

DHCP Server

In order for the DHCP server to provide configuration records for clients, an appropriate configuration file needs to be created. By default, the DHCP server looks for a configuration file called `dhcpd.conf` under `/etc`. You can either edit this file or create a new one and provide its full path to the DHCP server using the -cf flag. See a file example at `docs/dhcpd.conf` of the *Mellanox OFED for Linux* installation.

The DHCP server must run on a machine which has loaded the IPoIB module.

To run the DHCP server from the command line, enter:

```
dhcpd <IB network interface name> -d
```

Example:

```
host1# dhcpd ib0 -d
```

DHCP Client (Optional)

A DHCP client can be used if you need to prepare a diskless machine with an IB driver. See Step 8 under “Example: Adding an IB Driver to initrd (Linux)”.

In order to use a DHCP client identifier, you need to first create a configuration file that defines the DHCP client identifier. Then run the DHCP client with this file using the following command:

```
dhclient -cf <client conf file> <IB network interface name>
```

Example of a configuration file for the ConnectX (PCI Device ID 26428), called `dhclient.conf`:

```
# The value indicates a hexadecimal number
interface "ib1" {
send dhcp-client-identifier
ff:00:00:00:00:00:02:00:00:02:c9:00:00:02:c9:03:00:00:10:39;
}
```

Example of a configuration file for InfiniHost III Ex (PCI Device ID 25218), called `dhclient.conf`:

```
# The value indicates a hexadecimal number
interface "ib1" {
send dhcp-client-identifier
20:00:55:04:01:fe:80:00:00:00:00:00:00:00:02:c9:02:00:23:13:92;
}
```

In order to use the configuration file, run:

```
host1# dhclient -cf dhclient.conf ib1
```

4.6.3.2 Static IPoIB Configuration

If you wish to use an IPoIB configuration that is not based on DHCP, you need to supply the installation script with a configuration file (using the '-n' option) containing the full IP configuration. The IPoIB configuration file can specify either or both of the following data for an IPoIB interface:

- A static IPoIB configuration
- An IPoIB configuration based on an Ethernet configuration

  See your Linux distribution documentation for additional information about configuring IP addresses.

The following code lines are a template used to build the IPoIB configuration file running the install script:

```
# Static settings; all values provided by this file
IPADDR_ib0=11.4.3.175
NETMASK_ib0=255.255.0.0
NETWORK_ib0=11.4.0.0
BROADCAST_ib0=11.4.255.255
ONBOOT_ib0=1
# Based on eth0; each '*' will be replaced with a corresponding octet
# from eth0.
LAN_INTERFACE_ib0=eth0
IPADDR_ib0=11.4.'*'.'*'
NETMASK_ib0=255.255.0.0
NETWORK_ib0=11.4.0.0
```

```
BROADCAST_ib0=11.4.255.255
ONBOOT_ib0=1
# Based on the first eth<n> interface that is found (for n=0,1,...);
# each '*' will be replaced with a corresponding octet from eth<n>.
LAN_INTERFACE_ib0=
IPADDR_ib0=11.4.'*'.'*'
NETMASK_ib0=255.255.0.0
NETWORK_ib0=11.4.0.0
BROADCAST_ib0=11.4.255.255
ONBOOT_ib0=1
```

The parameters below must be added to the IPoIB interfaces' configuration files (ifcfg-ibX) on RHEL6.3 otherwise, different network interfaces may get the same IP address.

```
NM_CONTROLLED="yes"
TYPE="InfiniBand"
```

4.6.3.3 Manually Configuring IPoIB

This manual configuration persists only until the next reboot or driver restart.

To manually configure IPoIB for the default IB partition (VLAN), perform the following steps:

Step 1. To configure the interface, enter the **`ifconfig`** command with the following items:

- The appropriate IB interface (ib0, ib1, etc.)
- The IP address that you want to assign to the interface
- The netmask keyword
- The subnet mask that you want to assign to the interface

The following example shows how to configure an IB interface:

```
host1$ ifconfig ib0 11.4.3.175 netmask 255.255.0.0
```

Step 2. (Optional) Verify the configuration by entering the **`ifconfig`** command with the appropriate interface identifier *ib#* argument.

The following example shows how to verify the configuration:

```
host1$ ifconfig ib0
b0  Link encap:UNSPEC  HWaddr 80-00-04-04-FE-80-00-00-00-00-00-00-00-00-00-00
inet addr:11.4.3.175  Bcast:11.4.255.255  Mask:255.255.0.0
UP BROADCAST MULTICAST  MTU:65520  Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:128
RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

Step 3. Repeat Step 1 and Step 2 on the remaining interface(s).

4.6.4 Subinterfaces

You can create subinterfaces for a primary IPoIB interface to provide traffic isolation. Each such subinterface (also called a child interface) has a different IP and network addresses from the primary (parent) interface. The default Partition Key (PKey), ff:ff, applies to the primary (parent) interface.

This section describes how to

- Create a subinterface (Section 4.6.4.1)
- Remove a subinterface (Section 4.6.4.2)

4.6.4.1 Creating a Subinterface

In the following procedure, ib0 is used as an example of an IB subinterface.

To create a child interface (subinterface), follow this procedure:

Step 1. Decide on the PKey to be used in the subnet (valid values can be 0 or any 16-bit unsigned value). The actual PKey used is a 16-bit number with the most significant bit set. For example, a value of 0 will give a PKey with the value 0x8000.

Step 2. Create a child interface by running:

```
host1$ echo <PKey> > /sys/class/net/<IB subinterface>/create_child
```

Example:

```
host1$ echo 0 > /sys/class/net/ib0/create_child
```

This will create the interface ib0.8000.

Step 3. Verify the configuration of this interface by running:

```
host1$ ifconfig <subinterface>.<subinterface PKey>
```

Using the example of Step 2:

```
host1$ ifconfig ib0.8000
ib0.8000  Link encap:UNSPEC  HWaddr 80-00-00-4A-FE-80-00-00-00-00-00-00-00-00-00-00
BROADCAST MULTICAST  MTU:2044  Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:128
RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

Step 4. As can be seen, the interface does not have IP or network addresses. To configure those, you should follow the manual configuration procedure described in Section 4.6.3.3.

Step 5. To be able to use this interface, a configuration of the Subnet Manager is needed so that the PKey chosen, which defines a broadcast address, be recognized (see Chapter 9, "OpenSM – Subnet Manager").

4.6.4.2 Removing a Subinterface

To remove a child interface (subinterface), run:

```
echo <subinterface PKey> /sys/class/net/<ib_interface>/delete_child
```

Using the example of Step 2:

```
echo 0x8000 > /sys/class/net/ib0/delete_child
```

Note that when deleting the interface you must use the PKey value with the most significant bit set (e.g., 0x8000 in the example above).

4.6.5 Verifying IPoIB Functionality

To verify your configuration and your IPoIB functionality, perform the following steps:

Step 1. Verify the IPoIB functionality by using the **`ifconfig`** command.

The following example shows how two IB nodes are used to verify IPoIB functionality. In the following example, IB node 1 is at 11.4.3.175, and IB node 2 is at 11.4.3.176:

```
host1# ifconfig ib0 11.4.3.175 netmask 255.255.0.0
host2# ifconfig ib0 11.4.3.176 netmask 255.255.0.0
```

Step 2. Enter the ping command from 11.4.3.175 to 11.4.3.176.

The following example shows how to enter the ping command:

```
host1# ping -c 5 11.4.3.176
PING 11.4.3.176 (11.4.3.176) 56(84) bytes of data.
64 bytes from 11.4.3.176: icmp_seq=0 ttl=64 time=0.079 ms
64 bytes from 11.4.3.176: icmp_seq=1 ttl=64 time=0.044 ms
64 bytes from 11.4.3.176: icmp_seq=2 ttl=64 time=0.055 ms
64 bytes from 11.4.3.176: icmp_seq=3 ttl=64 time=0.049 ms
64 bytes from 11.4.3.176: icmp_seq=4 ttl=64 time=0.065 ms
--- 11.4.3.176 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3999ms rtt min/avg/max/mdev =
0.044/0.058/0.079/0.014 ms, pipe 2
```

4.6.6 Bonding IPoIB

To create an interface configuration script for the ibX and bondX interfaces, you should use the standard syntax (depending on your OS).

Bonding of IPoIB interfaces is accomplished in the same manner as would bonding of Ethernet interfaces: via the Linux Bonding Driver.

- Network Script files for IPoIB slaves are named after the IPoIB interfaces (e.g: ifcfg-ib0)

- The only meaningful bonding policy in IPoIB is High-Availability (bonding mode number 1, or active-backup)
- Bonding parameter "fail_over_mac" is meaningless in IPoIB interfaces, hence, the only supported value is the default: 0 (or "none" in SLES11)

For a persistent bonding IPoIB Network configuration, use the same Linux Network Scripts semantics, with the following exceptions/ additions:

- In the bonding master configuration file (e.g: ifcfg-bond0), in addition to Linux bonding semantics, use the following parameter: MTU=65520

65520 is a valid MTU value only if all IPoIB slaves operate in Connected mode (See Section 4.6.2, “IPoIB Mode Setting,” on page 79) and are configured with the same value. For IPoIB slaves that work in datagram mode, use MTU=2044. If you do not set the correct MTU or do not set MTU at all, performance of the interface might decrease.

- In the bonding slave configuration file (e.g: ifcfg-ib0), use the same Linux Network Scripts semantics. In particular: DEVICE=ib0
- In the bonding slave configuration file (e.g: ifcfg-ib0.8003), the line TYPE=InfiniBand is necessary when using bonding over devices configured with partitions (p_key)
- For RHEL users:

  In /etc/modprobe.b/bond.conf add the following lines :

```
alias bond0 bonding
```

- For SLES users:

  It is necessary to update the MANDATORY_DEVICES environment variable in /etc/sysconfig/network/config with the names of the IPoIB slave devices (e.g. ib0, ib1, etc.). Otherwise, bonding master may be created before IPoIB slave interfaces at boot time.

  It is possible to have multiple IPoIB bonding masters and a mix of IPoIB bonding master and Ethernet bonding master. However, It is NOT possible to mix Ethernet and IPoIB slaves under the same bonding master

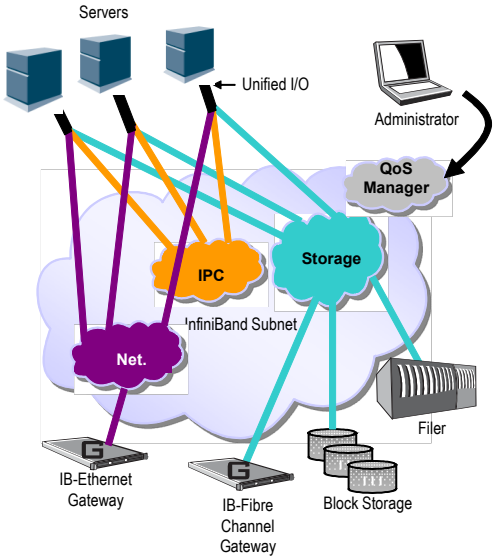
Restarting openibd does no keep the bonding configuration via Network Scripts. You have to restart the network service in order to bring up the bonding master. After the configuration is saved, restart the network service by running: /etc/init.d/network restart.

4.7 Quality of Service

4.7.1 Quality of Service Overview

Quality of Service (QoS) requirements stem from the realization of I/O consolidation over an IB network. As multiple applications and ULPs share the same fabric, a means is needed to control their use of network resources.

Figure 2: I/O Consolidation Over InfiniBand



QoS over Mellanox OFED for Linux is discussed in Chapter 9, “OpenSM – Subnet Manager”.

The basic need is to differentiate the service levels provided to different traffic flows, such that a policy can be enforced and can control each flow utilization of fabric resources.

The InfiniBand Architecture Specification defines several hardware features and management interfaces for supporting QoS:

- Up to 15 Virtual Lanes (VL) carry traffic in a non-blocking manner
- Arbitration between traffic of different VLs is performed by a two-priority-level weighted round robin arbiter. The arbiter is programmable with a sequence of (VL, weight) pairs and a maximal number of high priority credits to be processed before low priority is served
- Packets carry class of service marking in the range 0 to 15 in their header SL field
- Each switch can map the incoming packet by its SL to a particular output VL, based on a programmable table VL=SL-to-VL-MAP(in-port, out-port, SL)
- The Subnet Administrator controls the parameters of each communication flow by providing them as a response to Path Record (PR) or MultiPathRecord (MPR) queries

DiffServ architecture (IETF RFC 2474 & 2475) is widely used in highly dynamic fabrics. The following subsections provide the functional definition of the various software elements that enable a DiffServ-like architecture over the Mellanox OFED software stack.

4.7.2 QoS Architecture

QoS functionality is split between the SM/SA, CMA and the various ULPs. We take the “chronology approach” to describe how the overall system works.

1. The network manager (human) provides a set of rules (policy) that define how the network is being configured and how its resources are split to different QoS-Levels. The policy also define how to decide which QoS-Level each application or ULP or service use.
2. The SM analyzes the provided policy to see if it is realizable and performs the necessary fabric setup. Part of this policy defines the default QoS-Level of each partition. The SA is

enhanced to match the requested Source, Destination, QoS-Class, Service-ID, PKey against the policy, so clients (ULPs, programs) can obtain a policy enforced QoS. The SM may also set up partitions with appropriate IPoIB broadcast group. This broadcast group carries its QoS attributes: SL, MTU, RATE, and Packet Lifetime.

3. IPoIB is being setup. IPoIB uses the SL, MTU, RATE and Packet Lifetime available on the multicast group which forms the broadcast group of this partition.
4. MPI which provides non IB based connection management should be configured to run using hard coded SLs. It uses these SLs for every QP being opened.
5. ULPs that use CM interface (like SRP) have their own pre-assigned Service-ID and use it while obtaining PathRecord/MultiPathRecord (PR/MPR) for establishing connections. The SA receiving the PR/MPR matches it against the policy and returns the appropriate PR/MPR including SL, MTU, RATE and Lifetime.
6. ULPs and programs (e.g. SDP) use CMA to establish RC connection provide the CMA the target IP and port number. ULPs might also provide QoS-Class. The CMA then creates Service-ID for the ULP and passes this ID and optional QoS-Class in the PR/MPR request. The resulting PR/MPR is used for configuring the connection QP.

PathRecord and MultiPathRecord Enhancement for QoS:

As mentioned above, the PathRecord and MultiPathRecord attributes are enhanced to carry the Service-ID which is a 64bit value. A new field QoS-Class is also provided.

A new capability bit describes the SM QoS support in the SA class port info. This approach provides an easy migration path for existing access layer and ULPs by not introducing new set of PR/MPR attributes.

4.7.3 Supported Policy

The QoS policy, which is specified in a stand-alone file, is divided into the following four subsections:

I. Port Group

A set of CAs, Routers or Switches that share the same settings. A port group might be a partition defined by the partition manager policy, list of GUIDs, or list of port names based on NodeDescription.

II. Fabric Setup

Defines how the SL2VL and VLArb tables should be setup.

In OFED this part of the policy is ignored. SL2VL and VLArb tables should be configured in the OpenSM options file (opensm.opts).

III. QoS-Levels Definition

This section defines the possible sets of parameters for QoS that a client might be mapped to. Each set holds SL and optionally: Max MTU, Max Rate, Packet Lifetime and Path Bits.

Path Bits are not implemented in OFED.

IV. Matching Rules

A list of rules that match an incoming PR/MPR request to a QoS-Level. The rules are processed in order such as the first match is applied. Each rule is built out of a set of match expressions which should all match for the rule to apply. The matching expressions are defined for the following fields:

- SRC and DST to lists of port groups
- Service-ID to a list of Service-ID values or ranges
- QoS-Class to a list of QoS-Class values or ranges

4.7.4 CMA Features

The CMA interface supports Service-ID through the notion of port space as a prefix to the port number, which is part of the sockaddr provided to rdma_resolve_add(). The CMA also allows the ULP (like SDP) to propagate a request for a specific QoS-Class. The CMA uses the provided QoS-Class and Service-ID in the sent PR/MPR.

4.7.4.1 IPoIB

IPoIB queries the SA for its broadcast group information and uses the SL, MTU, RATE and Packet Lifetime available on the multicast group which forms this broadcast group.

4.7.4.2 SDP

SDP uses CMA for building its connections. The Service-ID for SDP is 0x000000000001PPPP, where PPPP are 4 hexadecimal digits holding the remote TCP/IP Port Number to connect to.

4.7.4.3 RDS

RDS uses CMA and thus it is very close to SDP. The Service-ID for RDS is 0x000000000106PPPP, where PPPP are 4 hexadecimal digits holding the TCP/IP Port Number that the protocol connects to.

The default port number for RDS is 0x48CA, which makes a default Service-ID 0x00000000010648CA.

4.7.4.4 SRP

The current SRP implementation uses its own CM callbacks (not CMA). So SRP fills in the Service-ID in the PR/MPR by itself and use that information in setting up the QP.

SRP Service-ID is defined by the SRP target I/O Controller (it also complies with IBTA Service-ID rules). The Service-ID is reported by the I/O Controller in the ServiceEntries DMA attribute and should be used in the PR/MPR if the SA reports its ability to handle QoS PR/MPRs.

4.7.5 OpenSM Features

The QoS related functionality that is provided by OpenSM—the Subnet Manager described in Chapter 9—can be split into two main parts:

I. Fabric Setup

During fabric initialization, the Subnet Manager parses the policy and apply its settings to the discovered fabric elements.

II. PR/MPR Query Handling

OpenSM enforces the provided policy on client request. The overall flow for such requests is: first the request is matched against the defined match rules such that the target QoS-Level definition is found. Given the QoS-Level a path(s) search is performed with the given restrictions imposed by that level.

4.8 Atomic Operations

4.8.1 Enhanced Atomic Operations

ConnectX® implements a set of Extended Atomic Operations beyond those defined by the IB spec. Atomicity guarantees, Atomic Ack generation, ordering rules and error behavior for this set of extended Atomic operations is the same as that for IB standard Atomic operations (as defined in section 9.4.5 of the IB spec).

4.8.1.1 Masked Compare and Swap (MskCmpSwap)

The MskCmpSwap atomic operation is an extension to the CmpSwap operation defined in the IB spec. MskCmpSwap allows the user to select a portion of the 64 bit target data for the "compare" check as well as to restrict the swap to a (possibly different) portion. The pseudocode below describes the operation:

```
| atomic_response = *va
| if (!((compare_add ^ *va) & compare_add_mask)) then
|     *va = (*va & ~(swap_mask)) | (swap & swap_mask)
|
| return atomic_response
```

The additional operands are carried in the Extended Transport Header. Atomic response generation and packet format for MskCmpSwap is as for standard IB Atomic operations.

4.8.1.2 Masked Fetch and Add (MFetchAdd)

The MFetchAdd Atomic operation extends the functionality of the standard IB FetchAdd by allowing the user to split the target into multiple fields of selectable length. The atomic add is done independently on each one of this fields. A bit set in the field_boundary parameter specifies the field boundaries. The pseudocode below describes the operation:

```
| bit_adder(ci, b1, b2, *co)
| {
```

```
|   value = ci + b1 + b2
|   *co = !!(value & 2)
|
|   return value & 1
| }
|
| #define MASK_IS_SET(mask, attr)      (!!((mask)&(attr)))
| bit_position = 1
| carry = 0
| atomic_response = 0
|
| for i = 0 to 63
| {
|         if ( i != 0 )
|                 bit_position =  bit_position << 1
|
|         bit_add_res = bit_adder(carry, MASK_IS_SET(*va, bit_position),
|                           MASK_IS_SET(compare_add, bit_position), &new_carry)
|         if (bit_add_res)
|                 atomic_response |= bit_position
|
|         carry = ((new_carry) && (!MASK_IS_SET(compare_add_mask, bit_position)))
| }
|
| return atomic_response
```

4.9 Huge Pages Support for Queue Resources

Buffer resources for QPs and CQs can now be set to use huge pages. When using huge pages, the HCA needs less MTT resources, thus improving performance by experiencing less cache misses. Huge pages are supported for:

- UD QPs
- RC QPs
- CQs

Huge pages are OFF by default. An application can be instructed to use huge pages by exporting to following environment variables:

- HUGE_UD=y
- HUGE_RC=y
- HUGE_CQ=y

For huge pages allocation to succeed, the system administrator will have to reserve huge pages from the OS. This can be done at runtime by running:

```
echo <nr_pages> > /proc/sys/vm/nr_hugepages to reserve nr_pages
```

If the system memory is too fragmented, the operation may fail. Therefore, we recommend performing this action after rebooting the system.

Since we are using IPC shared memory for allocating huge pages, occasionally, resources might not be freed. To delete old unused shared memory resources, use **`ipcrm`**.

5 VPI Configuration and Management

VPI allows ConnectX® ports to be independently configured as either IB or Eth.

5.1 Port Type Management

ConnectX® ports can be individually configured to work as InfiniBand or Ethernet ports. By default both ConnectX ports are initialized as InfiniBand ports. If you wish to change the port type use the `connectx_port_config` script after the driver is loaded.

When changing port type using the `"connectx_port_config"` utility, all the HCA's ports and interfaces are brought down, and then brought back up with a new port configuration.

Running "`/sbin/connectx_port_config -s`" will show current port configuration for all ConnectX devices.

Port configuration is saved in the file: `/etc/infiniband/connectx.conf`. This saved configuration is restored at driver restart only if restarting via "`/etc/init.d/openibd restart`".

Possible port types are:

- eth – Ethernet
- ib – Infiniband
- auto – Link sensing mode - Detect port type based on the attached network type. If no link is detected, the driver retries link sensing every few seconds.

Table 4 lists the ConnectX port configurations supported by VPI.

Table 4 - Supported ConnectX Port Configurations

Port 1 Configuration	Port 2 Configuration
ib	ib
ib	eth
eth	eth

Note that the configuration *Port1 = eth* and *Port2 = ib* is ***not*** supported.

The port link type can be configured for each device in the system at run time using the "`/sbin/connectx_port_config`" script. This utility will prompt for the PCI device to be modified (if there is only one it will be selected automatically).

In the next stage the user will be prompted for the desired mode for each port. The desired port configuration will then be set for the selected device.

This utility also has a non-interactive mode:

```
/sbin/connectx_port_config [[-d|--device <PCI device ID>] -c|--conf <port1,port2>]"
```

5.2 Auto Sensing

Auto Sensing enables the NIC to automatically sense the link type (InfiniBand or Ethernet) based on the link partner and load the appropriate driver stack (InfiniBand or Ethernet).

For example, if the first port is connected to an InfiniBand switch and the second to Ethernet switch, the NIC will automatically load the first switch as InfiniBand and the second as Ethernet.

5.2.1 Enabling Auto Sensing

Upon driver start up:

1. Sense the adapter card's port type:

   If a valid cable or module is connected (QSFP, SFP+, or SFP with EEPROM in the cable/module):

   - Set the port type to the sensed link type (IB/Ethernet)

   Otherwise:

   - Set the port type as default (Ethernet)

During driver run time:

- Sense a link every 3 seconds if no link is sensed/detected
- If sensed, set the port type as sensed

5.3 InfiniBand Driver

The InfiniBand driver, `mlx4_ib`, handles InfiniBand-specific functions and plugs into the InfiniBand midlayer.

5.4 Ethernet Driver

5.4.1 Overview

MLNX_EN driver is composed from mlx4_core and mlx4_en kernel modules and, exposes the following ConnectX® family capabilities:

- Single/Dual port
- Up to 16 Rx queues per port
- 16 Tx queues per port
- Rx steering mode: Receive Core Affinity (RCA)
- Tx arbitration mode: VLAN user-priority (off by default)
- MSI-X or INTx
- Adaptive interrupt moderation
- HW Tx/Rx checksum calculation
- Large Send Offload (i.e., TCP Segmentation Offload)
- Large Receive Offload
- Multi-core NAPI support
- VLAN Tx/Rx acceleration (HW VLAN stripping/insertion)

- HW VLAN filtering
- HW multicast filtering
- ifconfig up/down + mtu changes (up to 10K)
- Ethtool support
- Net device statistics
- CX4, QSFP and SFP+ connectors
- Wake-on-Lan support
- Ethernet - 10/40GigE

The current version of MLNX_OFED supports NC-SI in Ethernet mode only.

5.4.2 Loading the Ethernet Driver

By default, the Mellanox OFED stack loads `mlx4_en`. Run 'ifconfig -a' to verify that the module is listed.

5.4.3 Unloading the Driver

If `/etc/infiniband/openib.conf had` MLX4_EN_LOAD=yes at driver start-up, then you can unload the `mlx4_en` driver by running: `/etc/init.d/openibd stop`

Otherwise, unload `mlx4_en` by running:

```
#> modprobe -r mlx4_en
```

5.4.4 Ethernet Driver Usage and Configuration

- To assign an IP address to the interface run:

```
#> ifconfig eth<n> <ip>
```

where 'x' is the OS assigned interface number.

- To check driver and device information run:

```
#> ethtool -i eth<x>
```

Example:

```
ethtool -i eth2
driver: mlx4_en (MT_1020110019_CX-3)
version: 1.5.6.33 (Oct 2011)
firmware-version: 2.10.0000
bus-info: 0000:07:00.0
```

- To query stateless offload status run:

```
#> ethtool -k eth<x>
```

- To set stateless offload status run:

```
#> ethtool -K eth<x> [rx on|off] [tx on|off] [sg on|off] [tso on|off]
```

- To query interrupt coalescing settings run:

```
#> ethtool -c eth<x>
```

- By default, the driver uses adaptive interrupt moderation for the receive path, which adjusts the moderation time to the traffic pattern. To enable/disable adaptive interrupt moderation use the following command:

```
#>ethtool -C eth<x> adaptive-rx on|off
```

- Above an upper limit of packet rate, adaptive moderation will set the moderation time to its highest value. Below a lower limit of packet rate, the moderation time will be set to its lowest value. To set the values for packet rate limits and for moderation time high and low values, use the following command:

```
#> ethtool -C eth<x> [pkt-rate-low N] [pkt-rate-high N] [rx-usecs-low N] [rx-usecs-high
N]
```

- To set interrupt coalescing settings when adaptive moderation is disabled, use:

```
#> ethtool -c eth<x> [rx-usecs N] [rx-frames N]
```

usec settings correspond to the time to wait after the *last* packet is sent/received before triggering an interrupt.

- To query pause frame settings run:

```
#> ethtool -a eth<x>
```

- To set pause frame settings run:

```
#> ethtool -A eth<x> [rx on|off] [tx on|off]
```

- To query ring size values run:

```
#> ethtool -g eth<x>
```

- To modify rings size run:

```
#> ethtool -G eth<x> [rx <N>] [tx <N>]
```

- To obtain additional device statistics, run:

```
#> ethtool -S eth<x>
```

- To perform a self diagnostics test, run:

```
#> ethtool -t eth<x>
```

- The mlx4_en parameters can be found under `/sys/module/mlx4_en` (or `/sys/module/mlx4_en/parameters`, depending on the OS) and can be listed using the command:

```
#> modinfo mlx4_en
```

  To set non-default values to module parameters, the following line should be added to the file `/etc/modprobe.conf`:

```
"options mlx4_en <param_name>=<value> <param_name>=<value> ..."
```

6 Performance

6.1 General System Configurations

The following sections describe recommended configurations for system components and/or interfaces. Different systems may have different features, thus some recommendations below may not be applicable.

6.1.1 PCI Express (PCIe) Capabilities

Table 5 - Recommended PCIe Configuration

PCIe Generation	2.0
Speed	5GT/s
Width	x8
Max Payload size	256
Max Read Request	512

Note: For VPI / Ethernet adapters with ports configured to run 40Gb/s or above, it is recommended to use an x16 PCIe slot to benefit from the additional buffers allocated by the system.

6.1.2 BIOS Power Management Settings

- Set BIOS power management to “Maximum Performance”
- *On Intel Processors Only*: Disable C-states of PCI Express

Note that these performance optimizations may result in higher power consumption.

6.1.3 Intel® Hyper-Threading Technology

Hyper-Threading can increase message rate for multi process applications by having more logical cores. It might increase the latency of a single process, due to lower frequency of a single threaded core.

This section applies to Intel processors only supporting Hyper-Threading.

For latency and message rate sensitive applications, it is recommended to disable Hyper-Threading..

6.2 Improving IPv4/IPv6 Traffic Performance

You can use the Linux *sysctl* command to modify default system network parameters that are set by the operating system in order to improve IPv4 and IPv6 traffic performance. Note, however, that changing the network parameters may yield different results on different systems. The results are significantly dependent on the CPU and chipset efficiency.

6.2.1 Tuning the Network Adapter for Improved IPv4 Traffic Performance

The following changes are recommended for improving IPv4 traffic performance:

- Disable the TCP timestamps option for better CPU utilization:

```
sysctl -w net.ipv4.tcp_timestamps=0
```

- Disable the TCP selective acks option for better CPU utilization:

```
sysctl -w net.ipv4.tcp_sack=0
```

- Increase the maximum length of processor input queues:

```
sysctl -w net.core.netdev_max_backlog=250000
```

- Increase the TCP maximum and default buffer sizes using setsockopt():

```
sysctl  -w net.core.rmem_max=16777216
sysctl  -w net.core.wmem_max=16777216
sysctl  -w net.core.rmem_default=16777216
sysctl  -w net.core.wmem_default=16777216
sysctl  -w net.core.optmem_max=16777216
```

- Increase Linux's auto-tuning of TCP buffer limits. The minimum, default, and maximum number of bytes to use are:

```
sysctl -w net.ipv4.tcp_rmem="4096 87380 16777216"
sysctl -w net.ipv4.tcp_wmem="4096 65536 16777216"
```

6.2.2 Tuning the Network Adapter for Improved IPv6 Traffic Performance

The following changes are recommended for improving IPv6 traffic performance:

- Disable the TCP timestamps option for better CPU utilization:

```
sysctl -w net.ipv4.tcp_timestamps=0
```

- Disable the TCP selective acks option for better CPU utilization:

```
sysctl -w net.ipv4.tcp_sack=0
```

6.2.3 Interrupt Moderation

Interrupt moderation is used to decrease the frequency of network adapter interrupts to the CPU. Mellanox network adapters use an adaptive interrupt moderation algorithm by default. The algo-

rithm checks the transmission (Tx) and receive (Rx) packet rates and modifies the Rx interrupt moderation settings accordingly.

To manually set Tx and/or Rx interrupt moderation, use the ethtool utility. For example, the following commands first show the current (default) setting of interrupt moderation on the interface eth1, then turns off Rx interrupt moderation, and last shows the new setting.

```
> ethtool -c eth1
Coalesce parameters for eth1:
Adaptive RX: on  TX: off
...
pkt-rate-low: 400000
pkt-rate-high: 450000

rx-usecs: 16
rx-frames: 88
rx-usecs-irq: 0
rx-frames-irq: 0
...

> ethtool -C eth1 adaptive-rx off rx-usecs 0 rx-frames 0

> ethtool -c eth1
Coalesce parameters for eth1:
Adaptive RX: off TX: off
...
pkt-rate-low: 400000
pkt-rate-high: 450000

rx-usecs: 0
rx-frames: 0
rx-usecs-irq: 0
rx-frames-irq: 0
...
```

6.2.4 IRQ Affinity

The affinity of an interrupt is defined as the set of processor cores that service that interrupt. To improve application scalability and latency, it is recommended to distribute interrupt requests (IRQs) between the available processor cores.

The irqbalance service is running by default, and can set the IRQ affinity for the network interfaces anytime. However, on rare occasions, it might set the IRQ affinity to the remote NUMA node wrongfully.

➢ ***To stop the service and set the affinity manually:***

1. Stop irqbalance.

```
> service irqbalance stop
```

2. Set the IRQ affinity.

```
> echo <mask> > /proc/irq/<IRQ number>/smp_affinity
```

The following command turns off the IRQ balancer in RedHat:

```
> /etc/init.d/irqbalance stop
```

The following command turns off the IRQ balancer in SLES:

```
> /etc/init.d/irq_balancer stop
```

The following command assigns the affinity of a single interrupt vector:

```
> echo <hexadecimal bit mask> > /proc/irq/<irq vector>/smp_affinity
```

where bit *i* in <hexadecimal bit mask> indicates whether processor core *i* is in <irq vector>'s affinity or not.

6.2.4.1 Example: Script for Setting Interrupt Affinity

On systems that support NUMA, it is recommended to set IRQs from different network devices to processor cores that reside on different physical CPU sockets.

```
#! /bin/bash
CORES=$((`cat /proc/cpuinfo | grep processor | tail -1 | awk '{print $3}'`+1))
limit=1
while [ $CORES -gt 0 ]
do
        limit=$((limit*2))
        CORES=$((CORES-1))
done
if [ -z $1 ]; then
        IRQS=$(cat /proc/interrupts | grep eth-mlx | awk '{print $1}' |        sed 's/
://')
else
        IRQS=$(cat /proc/interrupts | grep $1 | awk '{print $1}' | sed 's/://')
fi
echo Discovered irqs: $IRQS
mask=1 ; for IRQ in $IRQS ; do echo $(printf "%x" $mask) > /proc/irq/$IRQ/smp_affinity
; mask=$(( mask * 2)) ; if [ $mask -ge $limit ] ; then mask=1 ; fi ; done
echo irqs were set OK.
```

6.2.5 Preserving Your Performance Settings After A Reboot

To preserve your performance settings after a reboot, you need to add them to the file /etc/sysctl.conf as follows:

```
<sysctl name1>=<value1>
<sysctl name2>=<value2>
<sysctl name3>=<value3>
```

```
<sysctl name4>=<value4>
```

For example, Section 6.2.1, “Tuning the Network Adapter for Improved IPv4 Traffic Performance” listed the following setting to disable the TCP timestamps option:

```
sysctl -w net.ipv4.tcp_timestamps=0
```

In order to keep the TCP timestamps option disabled after a reboot, add the following line to /etc/sysctl.conf:

```
net.ipv4.tcp_timestamps=0
```

6.3 Performance Troubleshooting

6.3.1 PCI Express Performance Troubleshooting

For the best performance on the PCI Express interface, the adapter card should be installed in an x8 slot with the following BIOS configuration parameters:

- Max_Read_Req, the maximum read request size, is 512 or higher
- MaxPayloadSize, the maximum payload size, is 256 or higher

A Max_Read_Req of 128 and/or installing the card in an x4 slot will significantly limit bandwidth.

To obtain the current setting for Max_Read_Req, enter:

```
setpci -d "15b3:" 68.w
```

To obtain the PCI Express slot (link) width and speed, enter:

```
setpci -d "15b3:" 72.B
```

1. If the output is neither 81 nor 82 card, then the card is NOT installed in an x8 PCI Express slot.
2. The least significant digit indicates the link speed:
   - 1 for PCI Express Gen 1 (2.5 GT/s)
   - 2 for PCI Express Gen 2 (5 GT/s)

If you are running InfiniBand at QDR (40Gb/s 4X IB ports), you must run PCI Express Gen 2.

6.3.2 InfiniBand Performance Troubleshooting

InfiniBand (IB) performance depends on the health of IB link(s) and on the IB card type. IB link speed (10Gb/s or SDR, 20Gb/s or DDR, 40Gb/s or QDR, 56Gb/s or FDR) also affects performance.

A latency sensitive application should take into account that each switch on the path adds ~200nsec at SDR, and 150nsec for DDR.

1. To check the IB link speed, enter:

```
ibstat
```

   Check the value indicated after the "Rate:" string: 10 indicates SDR, 20 indicates DDR, and 40 indicates QDR, 56 indicates FDR.

2. Check that the link has NO symbol errors since these errors result in the re-transmission of packets, and therefore in bandwidth loss. This check should be conducted for each port after the driver is loaded. To check for symbol errors, enter:

```
cat /sys/class/infiniband/<device>/ports/1/counters/symbol_error
```

   The command above is performed on Port 1 of the device <device>. The output value should be 0 if no symbol errors were recorded.

3. Bandwidth is expected to vary between systems. It heavily depends on the chipset, memory, and CPU. Nevertheless, the full-wire speed should be achieved by the host.
   - With IB @ SDR, the expected unidirectional full-wire speed bandwidth is ~900MB/sec.
   - With IB @ DDR and PCI Express Gen 1, the expected unidirectional full-wire speed bandwidth is ~1400MB/sec.
   - With IB @ DDR and PCI Express Gen 2, the expected unidirectional full-wire speed bandwidth is ~1800MB/sec.
   - With IB @ QDR and PCI Express Gen 2, the expected unidirectional full-wire speed bandwidth is ~3000MB/sec.
   - With IB @ FDR and PCI Express Gen 3 the expected unidirectional full-wire speed bandwidth is ~~6000MB/sec.

   To check the adapter's maximum bandwidth, use the `ib_write_bw` utility.

   To check the adapter's latency, use the `ib_write_lat` utility.

The utilities ib_write_bw and ib_write_lat are installed as part of Mellanox OFED.

6.3.3 System Performance Troubleshooting

On some systems it is recommended to change the power-saving configuration in order to achieve better performance. This configuration is usually handled by the BIOS. Please contact the system vendor for more information.

7 MPI - Message Passing Interface

7.1 Overview

PGI compiler does not support RHEL6.0, thus MLNX_OFED v1.5.2 will not include openmpi and mvapich with PGI compiler on RHEL6.

Mellanox OFED for Linux includes the following MPI implementations over InfiniBand and RoCE:

- Open MPI – an open source MPI-2 implementation by the Open MPI Project
- OSU MVAPICH – an MPI-1 implementation by Ohio State University

These MPI implementations, along with MPI benchmark tests such as OSU BW/LAT, Intel MPI Benchmark, and Presta, are installed on your machine as part of the Mellanox OFED for Linux installation. Table 6 lists some useful MPI links.

Table 6 - Useful MPI Links

MPI Standard	http://www-unix.mcs.anl.gov/mpi
Open MPI	http://www.open-mpi.org
MVAPICH MPI	http://mvapich.cse.ohio-state.edu/
MPI Forum	http://www.mpi-forum.org

This chapter includes the following sections:

- "Prerequisites for Running MPI" (page 104)
- "MPI Selector - Which MPI Runs" (page 105)
- "Compiling MPI Applications" (page 106)

7.2 Prerequisites for Running MPI

For launching multiple MPI processes on multiple remote machines, the MPI standard provides a launcher program that requires automatic login (i.e., password-less) onto the remote machines. SSH (Secure Shell) is both a computer program and a network protocol that can be used for logging and running commands on remote computers and/or servers.

7.2.1 SSH Configuration

The following steps describe how to configure password-less access over SSH:

Step 1. Generate an ssh key on the initiator machine (host1).

```
host1$ ssh-keygen -t rsa
Generating public/private rsa key pair.
```

```
Enter file in which to save the key (/home/<username>/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/<username>/.ssh/id_rsa.
Your public key has been saved in /home/<username>/.ssh/id_rsa.pub.
The key fingerprint is:
38:1b:29:df:4f:08:00:4a:0e:50:0f:05:44:e7:9f:05 <username>@host1
```

Step 2. Check that the public and private keys have been generated.

```
host1$ cd /home/<username>/.ssh/
host1$ ls
host1$ ls -la
total 40
drwx------  2 root root 4096 Mar  5 04:57 .
drwxr-x--- 13 root root 4096 Mar  4 18:27 ..
-rw-------  1 root root 1675 Mar  5 04:57 id_rsa
-rw-r--r--  1 root root  404 Mar  5 04:57 id_rsa.pub
```

Step 3. Check the public key.

```
host1$ cat id_rsa.pub
ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEA1zVY8VBHQh9okZN7OA1ibUQ74RXm4zHeczyVxpYHaDPyDmqezbYMKrCIVzd1
0bH+ZkC0rpLYviU0oUHd3fvNTfMs0gcGg08PysUf+12FyYjira2P1xyg6mkHLGGqVutfEMmABZ3wNCUg6J2X3G/
uiuSWXeubZmbXcMrP/
w4IWByfH8ajwo6A5WioNbFZElbYeeNfPZf4UNcgMOAMWp64sL58tkt32F+RGmyLXQWZL27Synsn6dHpxMqBorXNC
0ZBe4kTnUqm63nQ2z1qVMdL9FrCma1xIOu9+SQJAjwONevaMzFKEHe7YHg6YrNfXunfdbEurzB524TpPcrodZlfC
Q== <username>@host1
```

Step 4. Now you need to add the public key to the `authorized_keys2` file on the *target* machine.

```
host1$ cat id_rsa.pub | xargs ssh host2 \"echo  >>/home/<username>/.ssh/
authorized_keys2"
                                                  <username>@host2's pass-
word:                // Enter password
host1$
```

For a local machine, simply add the key to `authorized_keys2`.

```
host1$ cat id_rsa.pub >> authorized_keys2
```

Step 5. Test.

```
host1$ ssh  host2  uname
Linux
```

7.3 MPI Selector - Which MPI Runs

Mellanox OFED contains a simple mechanism for system administrators and end-users to select which MPI implementation they want to use. The MPI selector functionality is not specific to

any MPI implementation; it can be used with any implementation that provides shell startup files that correctly set the environment for that MPI. The Mellanox OFED installer will automatically add MPI selector support for each MPI that it installs. Additional MPI's not known by the Mellanox OFED installer can be listed in the MPI selector; see the mpi-selector(1) man page for details.

Note that MPI selector only affects the default MPI environment for *future* shells. Specifically, if you use MPI selector to select MPI implementation ABC, this default selection will not take effect until you start a new shell (e.g., logout and login again). Other packages (such as environment modules) provide functionality that allows changing your environment to point to a new MPI implementation in the current shell. The MPI selector was not meant to duplicate or replace that functionality.

The MPI selector functionality can be invoked in one of two ways:

1. The mpi-selector-menu command.

   This command is a simple, menu-based program that allows the selection of the system-wide MPI (usually only settable by root) and a per-user MPI selection. It also shows what the current selections are. This command is recommended for all users.

2. The mpi-selector command.

   This command is a CLI-equivalent of the mpi-selector-menu, allowing for the same functionality as mpi-selector-menu but without the interactive menus and prompts. It is suitable for scripting.

7.4 Compiling MPI Applications

A valid Fortran compiler must be present in order to build the MVAPICH MPI stack and tests.

The following compilers are supported by Mellanox OFED's MVAPICH and Open MPI packages: Gcc, Intel and PGI. The install script prompts the user to choose the compiler with which to install the MVAPICH and Open MPI RPMs. Note that more than one compiler can be selected simultaneously, if desired.

Compiling MVAPICH Applications

Please refer to http://mvapich.cse.ohio-state.edu/support/mvapich_user_guide.html.

To review the default configuration of the installation, check the default configuration file:

```
/usr/mpi/<compiler>/mvapich-<mvapich-ver>/etc/mvapich.conf
```

Compiling Open MPI Applications

Please refer to http://www.open-mpi.org/faq/?category=mpi-apps.

8 MellanoX Messaging

MellanoX Messaging (MXM) provides enhancements to parallel communication libraries by fully utilizing the underlying networking infrastructure provided by Mellanox HCA/switch hardware. This includes a variety of enhancements that take advantage of Mellanox networking hardware including:

- Multiple transport support including RC, XRC and UD
- Proper management of HCA resources and memory structures
- Efficient memory registration
- One-sided communication semantics
- Connection management
- Receive side tag matching
- Intra-node shared memory communication

These enhancements significantly increase the scalability and performance of message communications in the network, alleviating bottlenecks within the parallel communication libraries.

8.1 Enabling MXM in OpenMPI

➢ ***To enable MXM in OpenMPI, please run the following:***

```
 % svn co http://svn.open-mpi.org/svn/ompi/branches/v1.5
 % cd v1.5
 % ./autogen.sh && ./configure --prefix=$PWD/install --enable-debug --with-
mxm=$MXM_HOME/install --with-openib
 % make all && make install
```

9 OpenSM – Subnet Manager

9.1 Overview

OpenSM is an InfiniBand compliant Subnet Manager (SM). It is provided as a fixed flow executable called *opensm*, accompanied by a testing application called *osmtest*. OpenSM implements an InfiniBand compliant SM according to the InfiniBand Architecture Specification chapters: Management Model (13), Subnet Management (14), and Subnet Administration (15).

9.2 opensm Description

opensm is an InfiniBand compliant Subnet Manager and Subnet Administrator that runs on top of the Mellanox OFED stack. **opensm** performs the InfiniBand specification's required tasks for initializing InfiniBand hardware. One SM must be running for each InfiniBand subnet.

opensm also provides an experimental version of a performance manager.

opensm defaults were designed to meet the common case usage on clusters with up to a few hundred nodes. Thus, in this default mode, **opensm** will scan the IB fabric, initialize it, and sweep occasionally for changes.

opensm attaches to a specific IB port on the local machine and configures only the fabric connected to it. (If the local machine has other IB ports, **opensm** will ignore the fabrics connected to those other ports). If no port is specified, **opensm** will select the first "best" available port. **opensm** can also present the available ports and prompt for a port number to attach to.

By default, the **opensm** run is logged to two files: `/var/log/messages` and `/var/log/opensm.log`. The first file will register only general major events, whereas the second file will include details of reported errors. All errors reported in this second file should be treated as indicators of IB fabric health issues. (Note that when a fatal and non-recoverable error occurs, **opensm** will exit.) Both log files should include the message "SUBNET UP" if **opensm** was able to setup the subnet correctly.

9.2.1 opensm Syntax

```
opensm [OPTIONS]
```

where OPTIONS are:

```
--version
          Prints OpenSM version and exits.

--config, -F <file-name>
          The name of the OpenSM config file. When not specified
          /etc/opensm/opensm.conf will be used (if exists).

--create-config, -c <file-name>
          OpenSM will dump its configuration to the specified file and exit.
          This is a way to generate OpenSM configuration file template.

--guid, -g <GUID in hex>
```

```
          This option specifies the local port GUID value
          with which OpenSM should bind.  OpenSM may be
          bound to 1 port at a time.
          If GUID given is 0, OpenSM displays a list
          of possible port GUIDs and waits for user input.
          Without -g, OpenSM tries to use the default port.

--lmc, -l <LMC>
          This option specifies the subnet's LMC value.
          The number of LIDs assigned to each port is 2^LMC.
          The LMC value must be in the range 0-7.
          LMC values > 0 allow multiple paths between ports.
          LMC values > 0 should only be used if the subnet
          topology actually provides multiple paths between
          ports, i.e. multiple interconnects between switches.
          Without -l, OpenSM defaults to LMC = 0, which allows
          one path between any two ports.

--priority, -p <PRIORITY>
          This option specifies the SM's PRIORITY.
          This will effect the handover cases, where master
          is chosen by priority and GUID.  Range goes
          from 0 (lowest priority) to 15 (highest).

--smkey, -k <SM_Key>
          This option specifies the SM's SM_Key (64 bits).
          This will effect SM authentication.
          Note that OpenSM version 3.2.1 and below used the
          default value '1' in a host byte order, it is fixed
          now but you may need this option to interoperate
          with old OpenSM running on a little endian machine.

--reassign_lids, -r
          This option causes OpenSM to reassign LIDs to all
          end nodes. Specifying -r on a running subnet
          may disrupt subnet traffic.
          Without -r, OpenSM attempts to preserve existing
          LID assignments resolving multiple use of same LID.

--routing_engine, -R <engine name>
          This option chooses routing engine(s) to use instead of default
          Min Hop algorithm.  Multiple routing engines can be specified
          separated by commas so that specific ordering of routing
          algorithms will be tried if earlier routing engines fail.
          If all configured routing engines fail, OpenSM will always
          attempt to route with Min Hop unless 'no_fallback' is
```

```
          included in the list of routing engines.
          Supported engines: updn, file, ftree, lash, dor, torus-2QoS

--no_default_routing
          This option prevents OpenSM from falling back to default
          routing if none of the provided engines was able to
          configure the subnet

--do_mesh_analysis
          This option enables additional analysis for the lash
          routing engine to precondition switch port assignments
          in regular cartesian meshes which may reduce the number
          of SLs required to give a deadlock free routing

--lash_start_vl <vl number>
          Sets the starting VL to use for the lash routing algorithm.
          Defaults to 0.
--sm_sl <sl number>
          Sets the SL to use to communicate with the SM/SA. Defaults to 0.

--connect_roots, -z
          This option enforces routing engines (up/down and
          fat-tree) to make connectivity between root switches
          and in this way be IBA compliant. In many cases,
          this can violate "pure" deadlock free algorithm, so
          use it carefully.

--ucast_cache, -A
          This option enables unicast routing cache to prevent
          routing recalculation (which is a heavy task in a
          large cluster) when there was no topology change
          detected during the heavy sweep, or when the topology
          change does not require new routing calculation,
          e.g. in case of host reboot.
          This option becomes very handy when the cluster size
          is thousands of nodes.

--lid_matrix_file, -M <file name>
          This option specifies the name of the lid matrix dump file
          from where switch lid matrices (min hops tables will be
          loaded.

--lfts_file, -U <file name>
          This option specifies the name of the LFTs file
          from where switch forwarding tables will be loaded.
```

```
--sadb_file, -S <file name>
          This option specifies the name of the SA DB dump file
          from where SA database will be loaded.

--root_guid_file, -a <path to file>
          Set the root nodes for the Up/Down or Fat-Tree routing
          algorithm to the guids provided in the given file (one
          to a line)

--cn_guid_file, -u <path to file>
          Set the compute nodes for the Fat-Tree routing algorithm
          to the guids provided in the given file (one to a line)

--io_guid_file, -G <path to file>
          Set the I/O nodes for the Fat-Tree routing algorithm
          to the guids provided in the given file (one to a line)

--max_reverse_hops, -H <hop_count>
          Set the max number of hops the wrong way around
          an I/O node is allowed to do (connectivity for I/O nodes on top swithces)

--ids_guid_file, -m <path to file>
          Name of the map file with set of the IDs which will be used
          by Up/Down routing algorithm instead of node GUIDs
          (format: <guid> <id> per line)

--guid_routing_order_file, -X <path to file>
          Set the order port guids will be routed for the MinHop
          and Up/Down routing algorithms to the guids provided in the
          given file (one to a line)

--torus_config <path to file>
          This option defines the file name for the extra configuration
          info needed for the torus-2QoS routing engine.   The default
          name is '/etc/opensm/torus-2QoS.conf'

--once, -o
          This option causes OpenSM to configure the subnet
          once, then exit.  Ports remain in the ACTIVE state.

--sweep, -s <interval>
          This option specifies the number of seconds between
          subnet sweeps.  Specifying -s 0 disables sweeping.
          Without -s, OpenSM defaults to a sweep interval of
          10 seconds.
```

```
--timeout, -t <milliseconds>
          This option specifies the time in milliseconds
          used for transaction timeouts.
          Specifying -t 0 disables timeouts.
          Without -t, OpenSM defaults to a timeout value of
          200 milliseconds.

--retries <number>
          This option specifies the number of retries used
          for transactions.
          Without --retries, OpenSM defaults to 3 retries
          for transactions.

--maxsmps, -n <number>
          This option specifies the number of VL15 SMP MADs
          allowed on the wire at any one time.
          Specifying --maxsmps 0 allows unlimited outstanding
          SMPs.
          Without --maxsmps, OpenSM defaults to a maximum of
          4 outstanding SMPs.

--console, -q [off|local]
          This option activates the OpenSM console (default off).

--ignore-guids, -i <equalize-ignore-guids-file>
          This option provides the means to define a set of ports
          (by guid) that will be ignored by the link load
          equalization algorithm.

--hop_weights_file, -w <path to file>
          This option provides the means to define a weighting
          factor per port for customizing the least weight
          hops for the routing.

--dimn_ports_file, -O <path to file>
          This option provides the means to define a mapping
          between ports and dimension (Order) for controlling
          Dimension Order Routing (DOR).

--honor_guid2lid, -x
          This option forces OpenSM to honor the guid2lid file,
          when it comes out of Standby state, if such file exists
          under OSM_CACHE_DIR, and is valid. By default, this is FALSE.

--log_file, -f <log-file-name>
          This option defines the log to be the given file.
```

```
          By default, the log goes to /var/log/opensm.log.
          For the log to go to standard output use -f stdout.

--log_limit, -L <size in MB>
          This option defines maximal log file size in MB. When
          specified the log file will be truncated upon reaching
          this limit.

--erase_log_file, -e
          This option will cause deletion of the log file
          (if it previously exists). By default, the log file
          is accumulative.

--Pconfig, -P <partition-config-file>
          This option defines the optional partition configuration file.
          The default name is '/etc/opensm/partitions.conf'.

--no_part_enforce, -N
          This option disables partition enforcement on switch external ports.

--ar
          This option enables Adaptive Routing Manager in OpenSM.

--ar_config_file <path to file>
          This option specifies the optional Adaptive Routing config file.
          The default name is '/etc/opensm/osm-ar.conf'.

--qos, -Q
          This option enables QoS setup.

--qos_policy_file, -Y <QoS-policy-file>
          This option defines the optional QoS policy file.
          The default name is '/etc/opensm/qos-policy.conf'.

--stay_on_fatal, -y
          This option will cause SM not to exit on fatal initialization
          issues: if SM discovers duplicated guids or 12x link with
          lane reversal badly configured.
          By default, the SM will exit on these errors.

--daemon, -B
          Run in daemon mode - OpenSM will run in the background.

--inactive, -I
           Start SM in inactive rather than normal init SM state.
```

```
--prefix_routes_file <path to file>
          This option specifies the prefix routes file.
          Prefix routes control how the SA responds to path record
          queries for off-subnet DGIDs.  Default file is:
              /etc/opensm/prefix-routes.conf

--consolidate_ipv6_snm_req
          Use shared MLID for IPv6 Solicited Node Multicast groups
          per MGID scope and P_Key.

--log_prefix <prefix text>
          Prefix to syslog messages from OpenSM.

--verbose, -v
          This option increases the log verbosity level.
          The -v option may be specified multiple times
          to further increase the verbosity level.
          See the -D option for more information about
          log verbosity.

--V, -V
          This option sets the maximum verbosity level and
          forces log flushing.
          The -V is equivalent to '-D 0xFF -d 2'.
          See the -D option for more information about
          log verbosity.

--D, -D <flags>
          This option sets the log verbosity level.
          A flags field must follow the -D option.
          A bit set/clear in the flags enables/disables a
          specific log level as follows:
          BIT    LOG LEVEL ENABLED
          ----   -----------------
          0x01 - ERROR (error messages)
          0x02 - INFO (basic messages, low volume)
          0x04 - VERBOSE (interesting stuff, moderate volume)
          0x08 - DEBUG (diagnostic, high volume)
          0x10 - FUNCS (function entry/exit, very high volume)
          0x20 - FRAMES (dumps all SMP and GMP frames)
          0x40 - ROUTING (dump FDB routing information)
          0x80 - currently unused.
          Without -D, OpenSM defaults to ERROR + INFO (0x3).
          Specifying -D 0 disables all messages.
          Specifying -D 0xFF enables all messages (see -V).
          High verbosity levels may require increasing
```

```
          the transaction timeout with the -t option.

--debug, -d <number>
          This option specifies a debug option.
          These options are not normally needed.
          The number following -d selects the debug
          option to enable as follows:
          OPT   Description
          ---    -----------------
          -d0  - Ignore other SM nodes
          -d1  - Force single threaded dispatching
          -d2  - Force log flushing after each log message
          -d3  - Disable multicast support
          -d10 - Put OpenSM in testability mode
          Without -d, no debug options are enabled

--help, -h, -?
          Display this usage info then exit.
```

9.2.2 Environment Variables

The following environment variables control `opensm` behavior:

- OSM_TMP_DIR

  Controls the directory in which the temporary files generated by opensm are created. These files are: `opensm-subnet.lst`, `opensm.fdbs`, and `opensm.mcfdbs`. By default, this directory is `/var/log`.

- OSM_CACHE_DIR

  `opensm` stores certain data to the disk such that subsequent runs are consistent. The default directory used is `/var/cache/opensm`. The following file is included in it:

  - `guid2lid` – stores the LID range assigned to each GUID

9.2.3 Signaling

When opensm receives a HUP signal, it starts a new heavy sweep as if a trap has been received or a topology change has been found.

Also, SIGUSR1 can be used to trigger a reopen of `/var/log/opensm.log` for logrotate purposes.

9.2.4 Running opensm

The defaults of **opensm** were designed to meet the common case usage on clusters with up to a few hundred nodes. Thus, in this default mode, **opensm** will scan the IB fabric, initialize it, and sweep occasionally for changes. To run **opensm** in the default mode, simply enter:

```
host1# opensm
```

Note that **opensm** needs to be run on at least one machine in an IB subnet.

By default, an **opensm** run is logged to two files: `/var/log/messages` and `/var/log/opensm.log`. The first file, `message`, registers only general major events; the second file, `opensm.log`, includes details of reported errors. All errors reported in `opensm.log` should be treated as indicators of IB fabric health. Both log files should include the message "SUBNET UP" if opensm was able to setup the subnet correctly.

If a fatal, non-recoverable error occurs, opensm exits.

9.2.4.1 Running OpenSM As Daemon

OpenSM can also run as daemon. To run OpenSM in this mode, enter:

```
host1# /etc/init.d/opensmd start
```

9.3 osmtest Description

osmtest is a test program for validating the InfiniBand Subnet Manager and Subnet Administrator. **osmtest** provides a test suite for **opensm**. It can create an inventory file of all available nodes, ports, and PathRecords, including all their fields. It can also verify the existing inventory with all the object fields, and matches it to a pre-saved one. See Section 9.3.2.

osmtest has the following test flows:

- Multicast Compliancy test
- Event Forwarding test
- Service Record registration test
- RMPP stress test
- Small SA Queries stress test

9.3.1 Syntax

```
osmtest [OPTIONS]
```

where OPTIONS are:

```
-f, --flow       This option directs osmtest to run a specific flow:
                 Flow  Description:
                 c = create an inventory file with all nodes, ports and
                  paths
                 a = run all validation tests (expecting an input
                  inventory)
                 v = only validate the given inventory file
                 s = run service registration, deregistration, and lease
                  test
                 e = run event forwarding test
```

```
                  f = flood the SA with queries according to the stress mode
                  m = multicast flow
                  q = QoS info: dump VLArb and SLtoVL tables
                  t = run trap 64/65 flow (this flow requires running of
                   external tool)
                     Default = all flows except QoS
-w, --wait        This option specifies the wait time for trap 64/65 in
                 seconds. It is used only when running -f t - the trap 64/
                 65 flow Default = 10 sec
-d, --debug       This  option  specifies  a  debug option.  These options
                are not normally needed.  The number  following  -d
                selects  the  debug option to enable as follows:
                  OPT   Description
                  ---   -----------------
                  -d0   Ignore other SM nodes
                  -d1   Force single threaded dispatching
                  -d2   Force log flushing after each log message
                  -d3   Disable multicast support
-m, --max_lid     This  option specifies the maximal LID number to be
                searched for during inventory file build (Default = 100)
-g, --guid        This option specifies the  local  port  GUID  value  with
                which OpenSM  should  bind.   OpenSM may be bound to
                 1 port at a time. If GUID given is 0, OpenSM displays  a
                  list  of  possible  port GUIDs  and waits for user input.
                Without -g, OpenSM tries to use the default port.
-p, --port        This option displays a menu of possible local port  GUID
                values with which osmtest could bind
-i, --inventory    This  option  specifies the name of the inventory file
                 Normally, osmtest expects to find an inventory file,
                 which osmtest uses to validate real-time information
                 received from the SA during testing. If -i  is  not
                 specified,  osmtest  defaults  to  the  file
                 osmtest.dat.See -c option for related information
-s, --stress      This option runs the specified stress test instead of the
                normal test suite Stress test options are as follows:
                  OPT    Description
                  ---    -----------------
                  -s1    Single-MAD response SA queries
                  -s2    Multi-MAD (RMPP) response SA queries
                  -s3    Multi-MAD (RMPP) Path Record SA queries
                  Without -s, stress testing is not performed
-M, --Multicast_ModeThis option specify length of Multicast test:
                  OPT    Description
                  ---    -----------------
                  -M1    Short Multicast Flow (default) - single mode
                  -M2    Short Multicast Flow - multiple mode
```

```
                  -M3    Long Multicast Flow - single mode
                  -M4    Long Multicast Flow - multiple mode
                Single mode - Osmtest is tested alone, with no other  apps
                  that interact with OpenSM MC
                  Multiple  mode  -  Could  be  run  with other apps using
               MC with OpenSM. Without -M, default flow testing is per
               formed
-t, --timeout    This option specifies the time in milliseconds used for
               transaction  timeouts.  Specifying -t 0 disables
               timeouts. Without -t, OpenSM defaults to a timeout value
               of 200 milliseconds.
-l, --log_file   This option defines the log to be the given  file.   By
                default the log goes to /var/log/osm.log.  For the log to
               go to standard output use -f stdout.
-v, --verbose    This option increases the log verbosity level.   The  -v
               option may  be  specified  multiple  times to further
               increase the verbosity level.  See the -vf option for
               more  information  about log verbosity.
-V               This  option  sets  the  maximum  verbosity level and
               forces log flushing.  The -V is equivalent to '-vf 0xFF -
               d 2'.  See the -vf option for more information about log
               verbosity.
-vf              This  option  sets  the log verbosity level.  A flags
               field must follow the -D option.  A bit set/clear in the
               flags enables/disables a specific log level as follows:
                  BIT    LOG LEVEL ENABLED
                  ----   ----------------
                  0x01 - ERROR (error messages)
                  0x02 - INFO (basic messages, low volume)
                  0x04 - VERBOSE (interesting stuff, moderate volume)
                  0x08 - DEBUG (diagnostic, high volume)
                  0x10 - FUNCS (function entry/exit, very high volume)
                  0x20 - FRAMES (dumps all SMP and GMP frames)
                  0x40 - ROUTING (dump FDB routing information)
                  0x80 - currently unused.
               Without  -vf,  osmtest defaults to ERROR + INFO (0x3)
               Specifying -vf 0 disables all messages Specifying
               -vf 0xFF enables all messages  (see -V) High verbosity
               levels may require increasing the transaction timeout
               with the -t option
-h, --help       Display this usage info then exit.
```

9.3.2 Running osmtest

To run **osmtest** in the default mode, simply enter:

```
host1# osmtest
```

The default mode runs all the flows except for the Quality of Service flow (see Section 9.6).

After installing **opensm** (and if the InfiniBand fabric is stable), it is recommended to run the following command in order to generate the inventory file:

```
host1# osmtest -f c
```

Immediately afterwards, run the following command to test **opensm**:

```
host1# osmtest -f a
```

Finally, it is recommended to occasionally run "**`osmtest -v`**" (with verbosity) to verify that nothing in the fabric has changed.

9.4 Partitions

OpenSM enables the configuration of partitions (PKeys) in an InfiniBand fabric. By default, OpenSM searches for the partitions configuration file under the name `/usr/etc/opensm/partitions.conf`. To change this filename, you can use **opensm** with the '--Pconfig' or '-P' flags.

The default partition is created by OpenSM unconditionally, even when a partition configuration file does not exist or cannot be accessed.

The default partition has a P_Key value of 0x7fff. The port out of which runs OpenSM is assigned full membership in the default partition. All other end-ports are assigned partial membership.

9.4.1 File Format

Notes:

- Line content followed after '#' character is comment and ignored by parser.

General File Format

```
<Partition Definition>:<PortGUIDs list> ;
```

Partition Definition:

```
[PartitionName][=PKey][,flag[=value]][,defmember=full|limited]
```

where

```
PartitionName     string, will be used with logging. When omitted, an           empty
string will be used.
PKey              P_Key value for this partition. Only low 15 bits will
be used. When omitted, P_Key will be autogenerated.
flag              used to indicate IPoIB capability of this partition.
defmember=full|limited
                  specifies default membership for port guid list. Default is limited.
```

Currently recognized flags are:

```
 ipoib             indicates that this partition may be used for IPoIB, as a
result IPoIB capable MC group will be created.
 rate=<val>        specifies rate for this IPoIB MC group (default is 3 (10GBps))
 mtu=<val>         specifies MTU for this IPoIB MC group (default is 4 (2048))
 sl=<val>          specifies SL for this IPoIB MC group (default is 0)
 scope=<val>       specifies scope for this IPoIB MC group (default is 2 (link local))
```

Note that values for rate, mtu, and scope should be specified as defined in the IBTA specification (for example, mtu=4 for 2048).

PortGUIDs list:

```
 PortGUID          GUID of partition member EndPort. Hexadecimal numbers should start
from 0x, decimal numbers are accepted too.
 full or limited   indicates full or limited membership for this port. When omitted (or
unrecognized) limited membership is assumed.
```

There are two useful keywords for PortGUID definition:

- 'ALL' means all end-ports in this subnet
- 'SELF' means subnet manager's port

An empty list means that there are no ports in this partition.

Notes:

- White space is permitted between delimiters ('=', ',',':',';').
- The line can be wrapped after ':' after a Partition Definition and between.
- A PartitionName *does not* need to be unique, but PKey *does* need to be unique.
- If a PKey is repeated then the associated partition configurations will be merged and the first PartitionName will be used (see also next note).
- It is possible to split a partition configuration in more than one definition, but then they PKey should be explicitly specified (otherwise different PKey values will be generated for those definitions).

Examples:

```
 Default=0x7fff : ALL, SELF=full ;
 NewPartition , ipoib : 0x123456=full, 0x3456789034=limi,  0x2134af2306;

 YetAnotherOne = 0x300 : SELF=full ;
 YetAnotherOne = 0x300 : ALL=limited ;

 ShareIO = 0x80 , defmember=full : 0x123451, 0x123452;
 # 0x123453, 0x123454 will be limited
 ShareIO = 0x80 : 0x123453, 0x123454, 0x123455=full;
 # 0x123456, 0x123457 will be limited
```

```
 ShareIO   =   0x80   :   defmember=limited   :   0x123456,   0x123457,
0x123458=full;
 ShareIO = 0x80 , defmember=full : 0x123459, 0x12345a;
 ShareIO   =   0x80   ,   defmember=full   :   0x12345b,   0x12345c=limited,
0x12345d;
```

The following rule is equivalent to how OpenSM used to run prior to the partition manager:

```
 Default=0x7fff,ipoib:ALL=full;
```

9.5 Routing Algorithms

OpenSM offers six routing engines:

1. "Min Hop Algorithm"

   Based on the minimum hops to each node where the path length is optimized.

2. "UPDN Algorithm"

   Based on the minimum hops to each node, but it is constrained to ranking rules. This algorithm should be chosen if the subnet is not a pure Fat Tree, and a deadlock may occur due to a loop in the subnet.

3. "Fat-tree Routing Algorithm"

   This algorithm optimizes routing for a congestion-free "shift" communication pattern. It should be chosen if a subnet is a symmetrical Fat Tree of various types, not just a K-ary-N-Tree: non-constant K, not fully staffed, and for any CBB ratio. Similar to UPDN, Fat Tree routing is constrained to ranking rules.

4. "LASH Routing Algorithm"

   Uses InfiniBand virtual layers (SL) to provide deadlock-free shortest-path routing while also distributing the paths between layers. LASH is an alternative deadlock-free, topology-agnostic routing algorithm to the non-minimal UPDN algorithm. It avoids the use of a potentially congested root node.

5. "DOR Routing Algorithm"

   Based on the Min Hop algorithm, but avoids port equalization except for redundant links between the same two switches. This provides deadlock free routes for hypercubes when the fabric is cabled as a hypercube and for meshes when cabled as a mesh.

6. "Torus-2QoS Routing Algorithm"

   Based on the DOR Unicast routing algorithm specialized for 2D/3D torus topologies. Torus-2QoS provides deadlock-free routing while supporting two quality of service (QoS) levels. Additionally,

   it can route around multiple failed fabric links or a single failed fabric switch without introducing deadlocks, and without changing path SLvalues granted before the failure.

   OpenSM provides an optional unicast routing cache (enabled by -A or --ucast_cache options). When enabled, unicast routing cache prevents routing recalculation (which is a heavy task in a large cluster) when there was no topology change detected during the heavy sweep, or when the topology change does not require new routing calculation, e.g. when one or more CAs/RTRs/leaf switches going down, or one or more of these nodes coming back after being down. A very common case that is handled by the unicast routing cache is host reboot, which otherwise would cause two full routing recalculations: one when the host goes down, and the other when the host comes back online.

OpenSM also supports a file method which can load routes from a table – see Modular Routing Engine below.

The basic routing algorithm is comprised of two stages:

1. MinHop matrix calculation. How many hops are required to get from each port to each LID? The algorithm to fill these tables is different if you run standard (min hop) or Up/Down. For standard routing, a "relaxation" algorithm is used to propagate min hop from every destination LID through neighbor switches. For Up/Down routing, a BFS from every target is used. The BFS tracks link direction (up or down) and avoid steps that will perform up after a down step was used.
2. Once MinHop matrices exist, each switch is visited and for each target LID a decision is made as to what port should be used to get to that LID. This step is common to standard and Up/Down routing. Each port has a counter counting the number of target LIDs going through it. When there are multiple alternative ports with same MinHop to a LID, the one with less previously assigned ports is selected.

   If LMC > 0, more checks are added. Within each group of LIDs assigned to same target port:

   a. Use only ports which have same MinHop
   b. First prefer the ones that go to different systemImageGuid (then the previous LID of the same LMC group)
   c. If none, prefer those which go through another NodeGuid
   d. Fall back to the number of paths method (if all go to same node).

9.5.1 Effect of Topology Changes

OpenSM will preserve existing routing in any case where there is no change in the fabric switches unless the -r (--reassign_lids) option is specified.

```
-r, --reassign_lids
```

```
                    This option causes OpenSM to reassign LIDs to all end nodes. Specify-
ing -r on a running subnet may disrupt subnet traffic. Without -r, OpenSM attempts to
preserve existing LID assignments resolving multiple use of same LID.
```

If a link is added or removed, OpenSM does not recalculate the routes that do not have to change. A route has to change if the port is no longer UP or no longer the MinHop. When routing changes are performed, the same algorithm for balancing the routes is invoked.

In the case of using the file based routing, any topology changes are currently ignored The 'file' routing engine just loads the LFTs from the file specified, with no reaction to real topology. Obviously, this will not be able to recheck LIDs (by GUID) for disconnected nodes, and LFTs for non-existent switches will be skipped. Multicast is not affected by 'file' routing engine (this uses min hop tables).

9.5.2 Min Hop Algorithm

The Min Hop algorithm is invoked by default if no routing algorithm is specified. It can also be invoked by specifying '-R minhop'.

The Min Hop algorithm is divided into two stages: computation of min-hop tables on every switch and LFT output port assignment. Link subscription is also equalized with the ability to override based on port GUID. The latter is supplied by:

```
 -i <equalize-ignore-guids-file>
 -ignore-guids <equalize-ignore-guids-file>
                    This option provides the means to define a set of ports (by guids)
that will be ignored by the link load equalization algorithm.
```

LMC awareness routes based on (remote) system or switch basis.

9.5.3 UPDN Algorithm

The UPDN algorithm is designed to prevent deadlocks from occurring in loops of the subnet. A loop-deadlock is a situation in which it is no longer possible to send data between any two hosts connected through the loop. As such, the UPDN routing algorithm should be used if the subnet is not a pure Fat Tree, and one of its loops may experience a deadlock (due, for example, to high pressure).

The UPDN algorithm is based on the following main stages:

1. Auto-detect root nodes - based on the CA hop length from any switch in the subnet, a statistical histogram is built for each switch (hop num vs number of occurrences). If the histogram reflects a specific column (higher than others) for a certain node, then it is marked as a root node. Since the algorithm is statistical, it may not find any root nodes. The list of the root nodes found by this auto-detect stage is used by the ranking process stage.

The user can override the node list manually

If this stage cannot find any root nodes, and the user did not specify a guid list file, OpenSM defaults back to the Min Hop routing algorithm.

2. Ranking process - All root switch nodes (found in stage 1) are assigned a rank of 0. Using the BFS algorithm, the rest of the switch nodes in the subnet are ranked incrementally. This ranking aids in the process of enforcing rules that ensure loop-free paths.
3. Min Hop Table setting - after ranking is done, a BFS algorithm is run from each (CA or switch) node in the subnet. During the BFS process, the FDB table of each switch node traversed by BFS is updated, in reference to the starting node, based on the ranking rules and guid values.

At the end of the process, the updated FDB tables ensure loop-free paths through the subnet.

Up/Down routing does not allow LID routing communication between switches that are located inside spine “switch systems”. The reason is that there is no way to allow a LID route between them that does not break the Up/Down rule. One ramification of this is that you cannot run SM on switches other than the leaf switches of the fabric.

9.5.3.1 UPDN Algorithm Usage

Activation through OpenSM

- Use '-R updn' option (instead of old '-u') to activate the UPDN algorithm.
- Use '-a <root_guid_file>' for adding an UPDN guid file that contains the root nodes for ranking. If the `-a' option is not used, OpenSM uses its auto-detect root nodes algorithm.

Notes on the guid list file:

1. A valid guid file specifies one guid in each line. Lines with an invalid format will be discarded.
2. The user should specify the root switch guids. However, it is also possible to specify CA guids; OpenSM will use the guid of the switch (if it exists) that connects the CA to the subnet as a root node.

9.5.4 Fat-tree Routing Algorithm

The fat-tree algorithm optimizes routing for "shift" communication pattern. It should be chosen if a subnet is a symmetrical or almost symmetrical fat-tree of various types. It supports not just K-ary-N-Trees, by handling for non-constant K, cases where not all leafs (CAs) are present, any Constant Bisectional Ratio (CBB)ratio. As in UPDN, fat-tree also prevents credit-loop-deadlocks.

If the root guid file is not provided ('`-a`' or '`--root_guid_file`' options), the topology has to be pure fat-tree that complies with the following rules:

- Tree rank should be between two and eight (inclusively)

- Switches of the same rank should have the same number of UP-going port groups[1], unless they are root switches, in which case the shouldn't have UP-going ports at all.
- Switches of the same rank should have the same number of DOWN-going port groups, unless they are leaf switches.
- Switches of the same rank should have the same number of ports in each UP-going port group.
- Switches of the same rank should have the same number of ports in each DOWN-going port group.
- All the CAs have to be at the same tree level (rank).

If the root guid file is provided, the topology does not have to be pure fat-tree, and it should only comply with the following rules:

- Tree rank should be between two and eight (inclusively)
- All the Compute Nodes[2] have to be at the same tree level (rank). Note that non-compute node CAs are allowed here to be at different tree ranks.

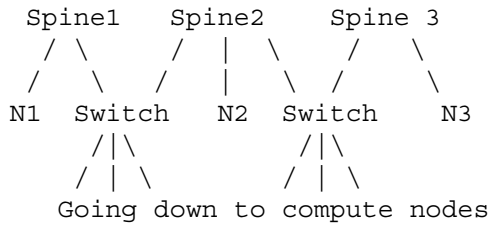
Topologies that do not comply cause a fallback to min hop routing. Note that this can also occur on link failures which cause the topology to no longer be a “pure” fat-tree.

Note that although fat-tree algorithm supports trees with non-integer CBB ratio, the routing will not be as balanced as in case of integer CBB ratio. In addition to this, although the algorithm allows leaf switches to have any number of CAs, the closer the tree is to be fully populated, the more effective the "shift" communication pattern will be. In general, even if the root list is provided, the closer the topology to a pure and symmetrical fat-tree, the more optimal the routing will be.

The algorithm also dumps compute node ordering file (`opensm-ftree-ca-order.dump`) in the same directory where the OpenSM log resides. This ordering file provides the CN order that may be used to create efficient communication pattern, that will match the routing tables.

9.5.4.1 Routing between non-CN Nodes

The use of the cn_guid_file option allows non-CN nodes to be located on different levels in the fat tree. In such case, it is not guaranteed that the Fat Tree algorithm will route between two non-CN nodes. In the scheme below, N1, N2 and N3 are non-CN nodes. Although all the CN have routes to and from them, there will not necessarily be a route between N1,N2 and N3. Such routes would require to use at least one of the switches the wrong way around.



1. Ports that are connected to the same remote switch are referenced as ‘port group’
2. List of compute nodes (CNs) can be specified by ‘-u’ or ‘--cn_guid_file’ OpenSM options.

To solve this problem, a list of non-CN nodes can be specified by \'-G\' or \'--io_guid_file\' option. These nodes will be allowed to use switches the wrong way around a specific number of times (specified by \'-H\' or \'--max_reverse_hops\'. With the proper max_reverse_hops and io_guid_file values, you can ensure full connectivity in the Fat Tree. In the scheme above, with a max_reverse_hop of 1, routes will be instanciated between N1<->N2 and N2<->N3. With a max_reverse_hops value of 2, N1,N2 and N3 will all have routes between them.

Using max_reverse_hops creates routes that use the switch in a counter-stream way. This option should never be used to connect nodes with high bandwidth traffic between them! It should only be used to allow connectivity for HA purposes or similar. Also having routes the other way around can cause credit loops.

9.5.4.2 Activation through OpenSM

- Use '-R ftree' option to activate the fat-tree algorithm

LMC > 0 is not supported by fat-tree routing. If this is specified, the default routing algorithm is invoked instead.

9.5.5 LASH Routing Algorithm

LASH is an acronym for LAyered SHortest Path Routing. It is a deterministic shortest path routing algorithm that enables topology agnostic deadlock-free routing within communication networks.

When computing the routing function, LASH analyzes the network topology for the shortest-path routes between all pairs of sources / destinations and groups these paths into virtual layers in such a way as to avoid deadlock.

LASH analyzes routes and ensures deadlock freedom between switch pairs. The link from HCA between and switch does not need virtual layers as deadlock will not arise between switch and HCA.

In more detail, the algorithm works as follows:

1. LASH determines the shortest-path between all pairs of source / destination switches. Note, LASH ensures the same SL is used for all SRC/DST - DST/SRC pairs and there is no guarantee that the return path for a given DST/SRC will be the reverse of the route SRC/DST.
2. LASH then begins an SL assignment process where a route is assigned to a layer (SL) if the addition of that route does not cause deadlock within that layer. This is achieved by maintaining and analysing a channel dependency graph for each layer. Once the potential addition of a path could lead to deadlock, LASH opens a new layer and continues the process.

3. Once this stage has been completed, it is highly likely that the first layers processed will contain more paths than the latter ones. To better balance the use of layers, LASH moves paths from one layer to another so that the number of paths in each layer averages out.

Note that the implementation of LASH in opensm attempts to use as few layers as possible. This number can be less than the number of actual layers available.

In general LASH is a very flexible algorithm. It can, for example, reduce to Dimension Order Routing in certain topologies, it is topology agnostic and fares well in the face of faults.

It has been shown that for both regular and irregular topologies, LASH outperforms Up/Down. The reason for this is that LASH distributes the traffic more evenly through a network, avoiding the bottleneck issues related to a root node and always routes shortest-path.

The algorithm was developed by Simula Research Laboratory.

Use '-R lash -Q' option to activate the LASH algorithm

QoS support has to be turned on in order that SL/VL mappings are used.

LMC > 0 is not supported by the LASH routing. If this is specified, the default routing algorithm is invoked instead.

For open regular cartesian meshes the DOR algorithm is the ideal routing algorithm. For toroidal meshes on the other hand there are routing loops that can cause deadlocks. LASH can be used to route these cases. The performance of LASH can be improved by preconditioning the mesh in cases where there are multiple links connecting switches and also in cases where the switches are not cabled consistently. To invoke this, use '-R lash -Q --do_mesh_analysis'. This will add an additional phase that analyses the mesh to try to determine the dimension and size of a mesh. If it determines that the mesh looks like an open or closed cartesian mesh it reorders the ports in dimension order before the rest of the LASH algorithm runs.

9.5.6 DOR Routing Algorithm

The Dimension Order Routing algorithm is based on the Min Hop algorithm and so uses shortest paths. Instead of spreading traffic out across different paths with the same shortest distance, it chooses among the available shortest paths based on an ordering of dimensions. Each port must be consistently cabled to represent a hypercube dimension or a mesh dimension. Paths are grown from a destination back to a source using the lowest dimension (port) of available paths at each step. This provides the ordering necessary to avoid deadlock. When there are multiple links between any two switches, they still represent only one dimension and traffic is balanced across them unless port equalization is turned off. In the case of hypercubes, the same port must be used throughout the fabric to represent the hypercube dimension and match on both ends of the cable. In the case of meshes, the dimension should consistently use the same pair of ports, one port on one end of the cable, and the other port on the other end, continuing along the mesh dimension.

Use '-R dor' option to activate the DOR algorithm.

9.5.7 Torus-2QoS Routing Algorithm

Torus-2QoS is a routing algorithm designed for large-scale 2D/3D torus fabrics. The torus-2QoS routing engine can provide the following functionality on a 2D/3D torus:

- Free of credit loops routing
- Two levels of QoS, assuming switches support 8 data VLs
- Ability to route around a single failed switch, and/or multiple failed links, without:
  - introducing credit loops
  - changing path SL values
- Very short run times, with good scaling properties as fabric size increases

9.5.7.1 Unicast Routing

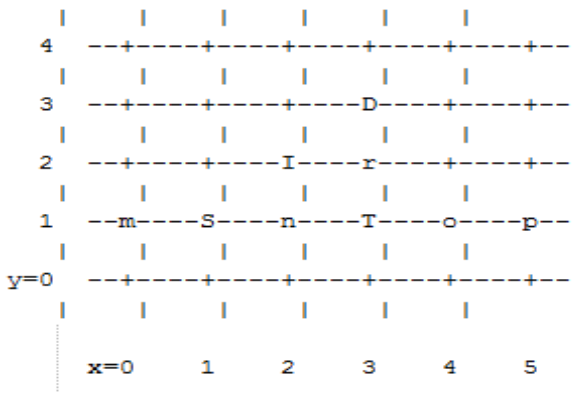
Torus-2QoS is a DOR-based algorithm that avoids deadlocks that would otherwise occur in a torus using the concept of a dateline for each torus dimension. It encodes into a path SL which datelines the path crosses as follows:

```
sl = 0;
for (d = 0; d < torus_dimensions; d++)
/* path_crosses_dateline(d) returns 0 or 1 */
sl |= path_crosses_dateline(d) << d;
```

For a 3D torus, that leaves one SL bit free, which torus-2QoS uses to implement two QoS levels. Torus-2QoS also makes use of the output port dependence of switch SL2VL maps to encode into one VL bit the information encoded in three SL bits. It computes in which torus coordinate direction each inter-switch link "points", and writes SL2VL maps for such ports as follows:

```
for (sl = 0; sl < 16; sl ++)
/* cdir(port) reports which torus coordinate direction a switch port
* "points" in, and returns 0, 1, or 2 */
sl2vl(iport,oport,sl) = 0x1 & (sl >> cdir(oport));
```

Thus, on a pristine 3D torus, i.e., in the absence of failed fabric switches, torus-2QoS consumes 8 SL values (SL bits 0-2) and 2 VL values (VL bit 0) per QoS level to provide deadlock-free routing on a 3D torus. Torus-2QoS routes around link failure by "taking the long way around" any 1D ring interrupted by a link failure. For example, consider the 2D 6x5 torus below, where switches are denoted by [+a-zA-Z]:



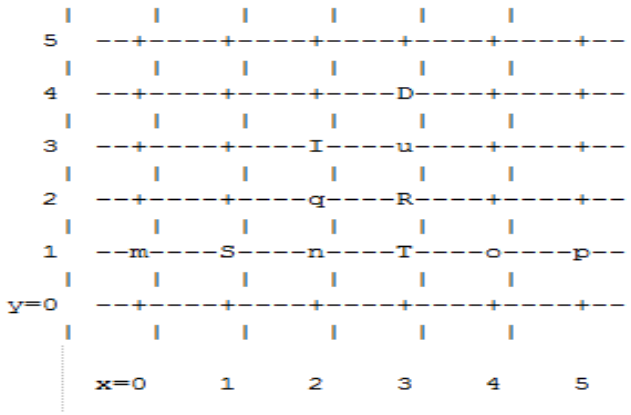
For a pristine fabric the path from S to D would be S-n-T-r-D. In the event that either link S-n or n-T has failed, torus-2QoS would use the path S-m-p-o-T-r-D.

Note that it can do this without changing the path SL value; once the 1D ring m-S-n-T-o-p-m has been broken by failure, path segments using it cannot contribute to deadlock, and the x-direction dateline (between, say, x=5 and x=0) can be ignored for path segments on that ring. One result of this is that torus-2QoS can route around many simultaneous link failures, as long as no 1D ring is broken into disjoint segments. For example, if links n-T and T-o have both failed, that ring has been broken into two disjoint segments, T and o-p-m-S-n. Torus-2QoS checks for such issues, reports if they are found, and refuses to route such fabrics.

Note that in the case where there are multiple parallel links between a pair of switches, torus-2QoS will allocate routes across such links in a round-robin fashion, based on ports at the path destination switch that are active and not used for inter-switch links. Should a link that is one of severalsuch parallel links fail, routes are redistributed across the remaining links. When the last of such a set of parallel links fails, traffic is rerouted as described above.

Handling a failed switch under DOR requires introducing into a path at least one turn that would be otherwise "illegal", i.e. not allowed by DOR rules. Torus-2QoS will introduce such a turn as close as possible to the failed switch in order to route around it. n the above example, suppose switch T has failed, and consider the path from S to D. Torus-2QoS will produce the path S-n-I-r-D, rather than the S-n-T-r-D path for a pristine torus, by introducing an early turn at n. Normal DOR rules will cause traffic arriving at switch I to be forwarded to switch r; for traffic arriving from I due to the "early" turn at n, this will generate an "illegal" turn at I.

Torus-2QoS will also use the input port dependence of SL2VL maps to set VL bit 1 (which would be otherwise unused) for y-x, z-x, and z-y turns, i.e., those turns that are illegal under DOR. This causes the first hop after any such turn to use a separate set of VL values, and prevents deadlock in the presence of a single failed switch. For any given path, only the hops after a turn that is illegal under DOR can contribute to a credit loop that leads to deadlock. So in the example above with failed switch T, the location of the illegal turn at I in the path from S to D requires that any credit loop caused by that turn must encircle the failed switch at T. Thus the second and later hops after the illegal turn at I (i.e., hop r-D) cannot contribute to a credit loop because they cannot be used to construct a loop encircling T. The hop I-r uses a separate VL, so it cannot contribute to a credit loop encircling T. Extending this argument shows that in addition to being capable of routing around a single switch failure without introducing deadlock, torus-2QoS can also route around multiple failed switches on the condition they are adjacent in the last dimension routed by DOR. For example, consider the following case on a 6x6 2D torus:

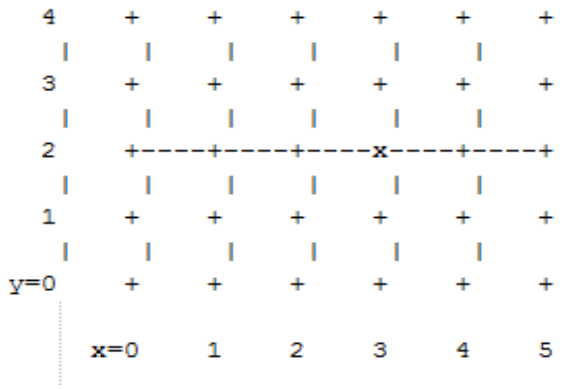


Suppose switches T and R have failed, and consider the path from S to D. Torus-2QoS will generate the path S-n-q-I-u-D, with an illegal turn at switch I, and with hop I-u using a VL with bit 1 set. As a further example, consider a case that torus-2QoS cannot route without deadlock: two failed switches adjacent in a dimension that is not the last dimension routed by DOR; here the failed switches are O and T:

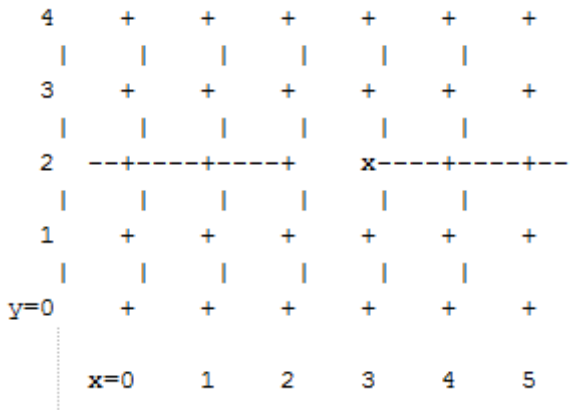
In a pristine fabric, torus-2QoS would generate the path from S to D as S-n-O-T-r-D. With failed switches O and T, torus-2QoS will generate the path S-n-I-q-r-D, with illegal turn at switch I, and with hop I-q using a VL with bit 1 set. In contrast to the earlier examples, the second hop after the illegal turn, q-r, can be used to construct a credit loop encircling the failed switches.

9.5.7.2 Multicast Routing

Since torus-2QoS uses all four available SL bits, and the three data VL bits that are typically available in current switches, there is no way to use SL/VL values to separate multicast traffic from unicast traffic. Thus, torus-2QoS must generate multicast routing such that credit loops cannot arise from a combination of multicast and unicast path segments. It turns out that it is possible to construct spanning trees for multicast routing that have that property. For the 2D 6x5 torus example above, here is the full-fabric spanning tree that torus-2QoS will construct, where "x" is the root switch and each "+" is a non-root switch:

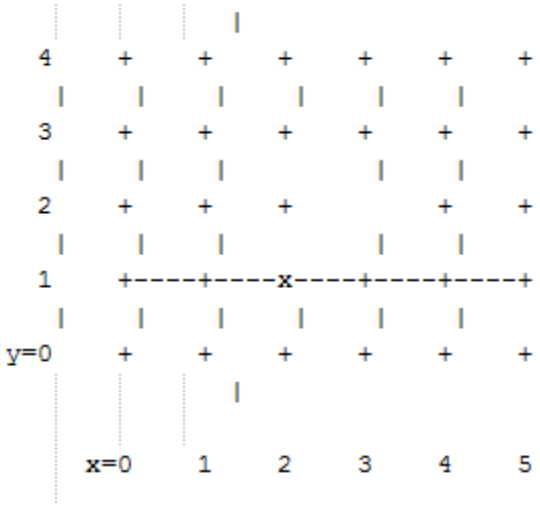


For multicast traffic routed from root to tip, every turn in the above spanning tree is a legal DOR turn. For traffic routed from tip to root, and some traffic routed through the root, turns are not legal DOR turns. However, to construct a credit loop, the union of multicast routing on this spanning tree with DOR unicast routing can only provide 3 of the 4 turns needed for the loop. In addition, if none of the above spanning tree branches crosses a dateline used for unicast credit loop avoidance on a torus, and if multicast traffic is confined to SL 0 or SL 8 (recall that torus-2QoS uses SL bit 3 to differentiate QoS level), then multicast traffic also cannot contribute to the "ring" credit loops that are otherwise possible in a torus. Torus-2QoS uses these ideas to create a master spanning tree. Every multicast group spanning tree will be constructed as a subset of the master tree, with the same root as the master tree. Such multicast group spanning trees will in general not be optimal for groups which are a subset of the full fabric. However, this compromise must be made to enable support for two QoS levels on a torus while preventing credit loops. In the presence of link or switch failures that result in a fabric for which torus-2QoS can generate credit-loop-free unicast routes, it is also possible to generate a master spanning tree for multicast that retains the required properties. For example, consider that same 2D 6x5 torus, with the link from (2,2) to (3,2) failed. Torus-2QoS will generate the following master spanning tree:



Two things are notable about this master spanning tree. First, assuming the x dateline was between x=5 and x=0, this spanning tree has a branch that crosses the dateline. However, just as for unicast, crossing a dateline on a 1D ring (here, the ring for y=2) that is broken by a failure cannot contribute to a torus credit loop. Second, this spanning tree is no longer optimal even for

multicast groups that encompass the entire fabric. That, unfortunately, is a compromise that must be made to retain the other desirable properties of torus-2QoS routing. In the event that a single switch fails, torus-2QoS will generate a master spanning tree that has no "extra" turns by appropriately selecting a root switch. In the 2D 6x5 torus example, assume now that the switch at (3,2), i.e. the root for a pristine fabric, fails. Torus-2QoS will generate the following master spanning tree for that case:



Assuming the y dateline was between y=4 and y=0, this spanning tree has a branch that crosses a dateline. However, again this cannot contribute to credit loops as it occurs on a 1D ring (the ring for x=3) that is broken by a failure, as in the above example.

9.5.7.3 Torus Topology Discovery

The algorithm used by torus-2QoS to construct the torus topology from the undirected graph representing the fabric requires that the radix of each dimension be configured via torus-2QoS.conf. It also requires that the torus topology be "seeded"; for a 3D torus this requires configuring four switches that define the three coordinate directions of the torus. Given this starting information, the algorithm is to examine the cube formed by the eight switch locations bounded by the corners (x,y,z) and (x+1,y+1,z+1). Based on switches already placed into the torus topology at some of these locations, the algorithm examines 4-loops of interswitch links to find the one that is consistent with a face of the cube of switch locations, and adds its swiches to the discovered topology in the correct locations.

Because the algorithm is based on examining the topology of 4-loops of links, a torus with one or more radix-4 dimensions requires extra initial seed configuration. See torus-2QoS.conf(5) for details. Torus-2QoS will detect and report when it has insufficient configuration for a torus with radix-4 dimensions.

In the event the torus is significantly degraded, i.e., there are many missing switches or links, it may happen that torus-2QoS is unable to place into the torus some switches and/or links that were discovered in the fabric, and will generate a warning in that case. A similar condition occurs if torus-2QoS is misconfigured, i.e., the radix of a torus dimension as configured does not match the radix of that torus dimension as wired, and many switches/links in the fabric will not be placed into the torus.

9.5.7.4 Quality Of Service Configuration

OpenSM will not program switchs and channel adapters with SL2VL maps or VL arbitration configuration unless it is invoked with -Q. Since torus-2QoS depends on such functionality for correct operation, always invoke OpenSM with -Q when torus-2QoS is in the list of routing engines. Any quality of service configuration method supported by OpenSM will work with torus-2QoS, subject to the following limitations and considerations. For all routing engines supported by OpenSM except torus-2QoS, there is a one-to-one correspondence between QoS level and SL. Torus-2QoS can only support two quality of service levels, so only the high-order bit of any SL value used for unicast QoS configuration will be honored by torus-2QoS. For multicast QoS configuration, only SL values 0 and 8 should be used with torus-2QoS.

Since SL to VL map configuration must be under the complete control of torus-2QoS, any configuration via qos_sl2vl, qos_swe_sl2vl, etc., must and will be ignored, and a warning will be generated. Torus-2QoS uses VL values 0-3 to implement one of its supported QoS levels, and VL values 4-7 to implement the other. Hard-to-diagnose application issues may arise if traffic is not delivered fairly across each of these two VL ranges. Torus-2QoS will detect and warn if VL arbitration is configured unfairly across VLs in the range 0-3, and also in the range 4-7. Note that the default OpenSM VL arbitration configuration does not meet this constraint, so all torus-2QoS users should configure VL arbitration via qos_vlarb_high, qos_vlarb_low, etc.

9.5.7.5 Operational Considerations

Any routing algorithm for a torus IB fabric must employ path SL values to avoid credit loops. As a result, all applications run over such fabrics must perform a path record query to obtain the correct path SL for connection setup. Applications that use rdma_cm for connection setup will automatically meet this requirement.

If a change in fabric topology causes changes in path SL values required to route without credit loops, in general all applications would need to repath to avoid message deadlock. Since torus-2QoS has the ability to reroute after a single switch failure without changing path SL values, repathing by running applications is not required when the fabric is routed with torus-2QoS.

Torus-2QoS can provide unchanging path SL values in the presence of subnet manager failover provided that all OpenSM instances have the same idea of dateline location. See torus-2QoS.conf(5) for details. Torus-2QoS will detect configurations of failed switches and links that prevent routing that is free of credit loops, and will log warnings and refuse to route. If "no_fallback" was configured in the list of OpenSM routing engines, then no other routing engine will attempt to route the fabric. In that case all paths that do not transit the failed components will continue to work, and the subset of paths that are still operational will continue to remain free of credit loops. OpenSM will continue to attempt to route the fabric after every sweep interval, and after any change (such as a link up) in the fabric topology. When the fabric components are repaired, full functionality will be restored. In the event OpenSM was configured to allow some other engine to route the fabric if torus-2QoS fails, then credit loops and message deadlock are likely if torus-2QoS had previously routed the fabric successfully. Even if the other engine is capable of routing a torus without credit loops, applications that built connections with path SL values granted under torus-2QoS will likely experience message deadlock under routing generated by a different engine, unless they repath. To verify that a torus fabric is routed free of credit loops, use ibdmchk to analyze data collected via ibdiagnet -vlr.

9.5.7.6 Torus-2QoS Configuration File Syntax

The file torus-2QoS.conf contains configuration information that is specific to the OpenSM routing engine torus-2QoS. Blank lines and lines where the first non-whitespace character is "#" are ignored. A token is any contiguous group of non-whitespace characters. Any tokens on a line following the recognized configuration tokens described below are ignored.

```
[torus|mesh] x_radix[m|M|t|T] y_radix[m|M|t|T] z_radix[m|M|t|T]
```

Either torus or mesh must be the first keyword in the configuration, and sets the topology that torus-2QoS will try to construct. A 2D topology can be configured by specifying one of x_radix, y_radix, or z_radix as 1. An individual dimension can be configured as mesh (open) or torus (looped) by suffixing its radix specification with one of m, M, t, or T. Thus, "mesh 3T 4 5" and "torus 3 4M 5M" both specify the same topology.

Note that although torus-2QoS can route mesh fabrics, its ability to route around failed components is severely compromised on such fabrics. A failed fabric componentis very likely to cause a disjoint ring; see UNICAST ROUTING in torus-2QoS(8).

```
xp_link sw0_GUID sw1_GUID
yp_link sw0_GUID sw1_GUID
zp_link sw0_GUID sw1_GUID
xm_link sw0_GUID sw1_GUID
ym_link sw0_GUID sw1_GUID
zm_link sw0_GUID sw1_GUID
```

These keywords are used to seed the torus/mesh topology. For example, "xp_link 0x2000 0x2001" specifies that a link from the switch with node GUID 0x2000 to the switch with node GUID 0x2001 would point in the positive x direction, while "xm_link 0x2000 0x2001" specifies that a link from the switch with node GUID 0x2000 to the switch with node GUID 0x2001 would point in the negative x direction. All the link keywords for a given seed must specify the same "from" switch.

In general, it is not necessary to configure both the positive and negative directions for a given coordinate; either is sufficient. However, the algorithm used for topology discovery needs extra information for torus dimensions of radix four (see TOPOLOGY DISCOVERY in torus-2QoS(8)). For such cases both the positive and negative coordinate directions must be specified.

Based on the topology specified via the torus/mesh keyword, torus-2QoS will detect and log when it has insufficient seed configuration.

```
x_dateline position
y_dateline position
z_dateline position
```

In order for torus-2QoS to provide the guarantee that path SL values do not change under any conditions for which it can still route the fabric, its idea of dateline position must not change relative to physical switch locations. The dateline keywords provide the means to configure such behavior.

The dateline for a torus dimension is always between the switch with coordinate 0 and the switch with coordinate radix-1 for that dimension. By default, the common switch in a torus seed is taken as the origin of the coordinate system used to describe switch location. The position param-

eter for a dateline keyword moves the origin (and hence the dateline) the specified amount relative to the common switch in a torus seed.

```
next_seed
```

If any of the switches used to specify a seed were to fail torus-2QoS would be unable to complete topology discovery successfully. The next_seed keyword specifies that the following link and dateline keywords apply to a new seed specification.

For maximum resiliency, no seed specification should share a switch with any other seed specification. Multiple seed specifications should use dateline configuration to ensure that torus-2QoS can grant path SL values that are constant, regardless of which seed was used to initiate topology discovery.

portgroup_max_ports max_ports - This keyword specifies the maximum number of parallel inter-switch links, and also the maximum number of host ports per switch, that torus-2QoS can accommodate. The default value is 16. Torus-2QoS will log an error message during topology discovery if this parameter needs to be increased. If this keyword appears multiple times, the last instance prevails.

port_order p1 p2 p3 ... - This keyword specifies the order in which CA ports on a destination switch are visited when computing routes. When the fabric contains switches connected with multiple parallel links, routes are distributed in a round-robin fashion across such links, and so changing the order that CA ports are visited changes the distribution of routes across such links. This may be advantageous for some specific traffic patterns.

The default is to visit CA ports in increasing port order on destination switches. Duplicate values in the list will be ignored.

EXAMPLE

```
# Look for a 2D (since x radix is one) 4x5 torus.
torus 1 4 5
# y is radix-4 torus dimension, need both
# ym_link and yp_link configuration.
yp_link 0x200000 0x200005 # sw @ y=0,z=0 -> sw @ y=1,z=0
ym_link 0x200000 0x20000f # sw @ y=0,z=0 -> sw @ y=3,z=0
# z is not radix-4 torus dimension, only need one of
# zm_link or zp_link configuration.
zp_link 0x200000 0x200001 # sw @ y=0,z=0 -> sw @ y=0,z=1
next_seed
yp_link 0x20000b 0x200010 # sw @ y=2,z=1 -> sw @ y=3,z=1
ym_link 0x20000b 0x200006 # sw @ y=2,z=1 -> sw @ y=1,z=1
zp_link 0x20000b 0x20000c # sw @ y=2,z=1 -> sw @ y=2,z=2
y_dateline -2 # Move the dateline for this seed
z_dateline -1 # back to its original position.
# If OpenSM failover is configured, for maximum resiliency
# one instance should run on a host attached to a switch
# from the first seed, and another instance should run
# on a host attached to a switch from the second seed.
# Both instances should use this torus-2QoS.conf to ensure
# path SL values do not change in the event of SM failover.
# port_order defines the order on which the ports would be
# chosen for routing.
port_order 7 10 8 11 9 12 25 28 26 29 27 30
```

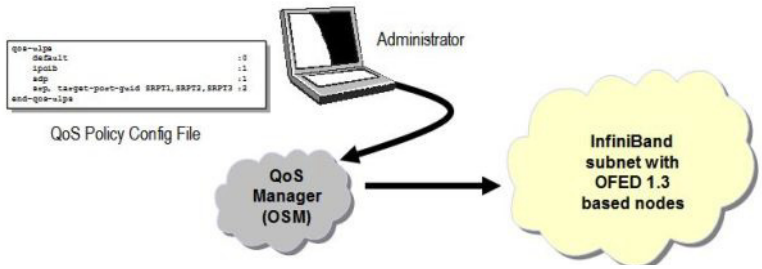
9.6 Quality of Service Management in OpenSM

9.6.1 Overview

When Quality of Service (QoS) in OpenSM is enabled (using the '-Q' or '--qos' flags), OpenSM looks for a QoS Policy file. During fabric initialization and at every heavy sweep, OpenSM parses the QoS policy file, applies its settings to the discovered fabric elements, and enforces the provided policy on client requests. The overall flow for such requests is as follows:

- The request is matched against the defined matching rules such that the QoS Level definition is found
- Given the QoS Level, a path(s) search is performed with the given restrictions imposed by that level

Figure 3: QoS Manager



There are two ways to define QoS policy:

- Advanced – the advanced policy file syntax provides the administrator various ways to match a PathRecord/MultiPathRecord (PR/MPR) request, and to enforce various QoS constraints on the requested PR/MPR
- Simple – the simple policy file syntax enables the administrator to match PR/MPR requests by various ULPs and applications running on top of these ULPs

9.6.2 Advanced QoS Policy File

The QoS policy file has the following sections:

I) Port Groups (denoted by port-groups)

This section defines zero or more port groups that can be referred later by matching rules (see below). Port group lists ports by:

- Port GUID
- Port name, which is a combination of NodeDescription and IB port number
- PKey, which means that all the ports in the subnet that belong to partition with a given PKey belong to this port group
- Partition name, which means that all the ports in the subnet that belong to partition with a given name belong to this port group
- Node type, where possible node types are: CA, SWITCH, ROUTER, ALL, and SELF (SM's port).

II) QoS Setup (denoted by qos-setup)

This section describes how to set up SL2VL and VL Arbitration tables on various nodes in the fabric. However, this is not supported in OFED. SL2VL and VLArb tables should be configured in the OpenSM options file (default location - /var/cache/opensm/opensm.opts).

III) QoS Levels (denoted by qos-levels)

Each QoS Level defines Service Level (SL) and a few optional fields:

- MTU limit
- Rate limit
- PKey
- Packet lifetime

When path(s) search is performed, it is done with regards to restriction that these QoS Level parameters impose. One QoS level that is mandatory to define is a DEFAULT QoS level. It is applied to a PR/MPR query that does not match any existing match rule. Similar to any other QoS Level, it can also be explicitly referred by any match rule.

IV) QoS Matching Rules (denoted by qos-match-rules)

Each PathRecord/MultiPathRecord query that OpenSM receives is matched against the set of matching rules. Rules are scanned in order of appearance in the QoS policy file such as the first match takes precedence.

Each rule has a name of QoS level that will be applied to the matching query. A default QoS level is applied to a query that did not match any rule.

Queries can be matched by:

- Source port group (whether a source port is a member of a specified group)
- Destination port group (same as above, only for destination port)
- PKey
- QoS class
- Service ID

To match a certain matching rule, PR/MPR query has to match ALL the rule's criteria. However, not all the fields of the PR/MPR query have to appear in the matching rule.

For instance, if the rule has a single criterion - Service ID, it will match any query that has this Service ID, disregarding rest of the query fields. However, if a certain query has only Service ID (which means that this is the only bit in the PR/MPR component mask that is on), it will not match any rule that has other matching criteria besides Service ID.

9.6.3 Simple QoS Policy Definition

Simple QoS policy definition comprises of a single section denoted by qos-ulps. Similar to the advanced QoS policy, it has a list of match rules and their QoS Level, but in this case a match rule has only one criterion - its goal is to match a certain ULP (or a certain application on top of this ULP) PR/MPR request, and QoS Level has only one constraint - Service Level (SL).

The simple policy section may appear in the policy file in combine with the advanced policy, or as a stand-alone policy definition. See more details and list of match rule criteria below.

9.6.4 Policy File Syntax Guidelines

- Leading and trailing blanks, as well as empty lines, are ignored, so the indentation in the example is just for better readability.
- Comments are started with the pound sign (#) and terminated by EOL.
- Any keyword should be the first non-blank in the line, unless it's a comment.
- Keywords that denote section/subsection start have matching closing keywords.
- Having a QoS Level named "DEFAULT" is a must - it is applied to PR/MPR requests that didn't match any of the matching rules.
- Any section/subsection of the policy file is optional.

9.6.5 Examples of Advanced Policy File

As mentioned earlier, any section of the policy file is optional, and the only mandatory part of the policy file is a default QoS Level.

Here's an example of the shortest policy file:

```
qos-levels
    qos-level
        name: DEFAULT
        sl: 0
    end-qos-level
end-qos-levels
```

Port groups section is missing because there are no match rules, which means that port groups are not referred anywhere, and there is no need defining them. And since this policy file doesn't have any matching rules, PR/MPR query will not match any rule, and OpenSM will enforce default QoS level. Essentially, the above example is equivalent to not having a QoS policy file at all.

The following example shows all the possible options and keywords in the policy file and their syntax:

```
#
# See the comments in the following example.
# They explain different keywords and their meaning.
#
port-groups

    port-group # using port GUIDs
        name: Storage
        # "use" is just a description that is used for logging
        #  Other than that, it is just a comment
        use: SRP Targets
        port-guid: 0x10000000000001, 0x10000000000005-0x1000000000FFFA
        port-guid: 0x1000000000FFFF
    end-port-group

    port-group
        name: Virtual Servers
        # The syntax of the port name is as follows:
        #   "node_description/Pnum".
        # node_description is compared to the NodeDescription of the node,
        # and "Pnum" is a port number on that node.
        port-name: vs1 HCA-1/P1, vs2 HCA-1/P1
    end-port-group

    # using partitions defined in the partition policy
    port-group
        name: Partitions
        partition: Part1
        pkey: 0x1234
    end-port-group

    # using node types: CA, ROUTER, SWITCH, SELF (for node that runs SM)
    # or ALL (for all the nodes in the subnet)
    port-group
        name: CAs and SM
        node-type: CA, SELF
    end-port-group
```

```
end-port-groups

qos-setup
    # This section of the policy file describes how to set up SL2VL and VL
    # Arbitration tables on various nodes in the fabric.
    # However, this is not supported in OFED - the section is parsed
    # and ignored. SL2VL and VLArb tables should be configured in the
    # OpenSM options file (by default - /var/cache/opensm/opensm.opts).
end-qos-setup

qos-levels

    # Having a QoS Level named "DEFAULT" is a must - it is applied to
    # PR/MPR requests that didn't match any of the matching rules.
    qos-level
        name: DEFAULT
        use: default QoS Level
        sl: 0
    end-qos-level

    # the whole set: SL, MTU-Limit, Rate-Limit, PKey, Packet Lifetime
    qos-level
        name: WholeSet
        sl: 1
        mtu-limit: 4
        rate-limit: 5
        pkey: 0x1234
        packet-life: 8
    end-qos-level

end-qos-levels

# Match rules are scanned in order of their apperance in the policy file.
# First matched rule takes precedence.
qos-match-rules

    # matching by single criteria: QoS class
    qos-match-rule
        use: by QoS class
        qos-class: 7-9,11
        # Name of qos-level to apply to the matching PR/MPR
        qos-level-name: WholeSet
    end-qos-match-rule

    # show matching by destination group and service id
    qos-match-rule
```

```
            use: Storage targets
            destination: Storage
            service-id: 0x10000000000001, 0x10000000000008-0x10000000000FFF
            qos-level-name: WholeSet
        end-qos-match-rule

        qos-match-rule
            source: Storage
            use: match by source group only
            qos-level-name: DEFAULT
        end-qos-match-rule

        qos-match-rule
            use: match by all parameters
            qos-class: 7-9,11
            source: Virtual Servers
            destination: Storage
            service-id: 0x0000000000010000-0x000000000001FFFF
            pkey: 0x0F00-0x0FFF
            qos-level-name: WholeSet
        end-qos-match-rule

    end-qos-match-rules
```

9.6.6 Simple QoS Policy - Details and Examples

Simple QoS policy match rules are tailored for matching ULPs (or some application on top of a ULP) PR/MPR requests. This section has a list of per-ULP (or per-application) match rules and the SL that should be enforced on the matched PR/MPR query.

Match rules include:

- Default match rule that is applied to PR/MPR query that didn't match any of the other match rules
- SDP
- SDP application with a specific target TCP/IP port range
- SRP with a specific target IB port GUID
- RDS
- IPoIB with a default PKey
- IPoIB with a specific PKey
- Any ULP/application with a specific Service ID in the PR/MPR query
- Any ULP/application with a specific PKey in the PR/MPR query
- Any ULP/application with a specific target IB port GUID in the PR/MPR query

Since any section of the policy file is optional, as long as basic rules of the file are kept (such as no referring to nonexisting port group, having default QoS Level, etc), the simple policy section (qos-ulps) can serve as a complete QoS policy file.

The shortest policy file in this case would be as follows:

```
qos-ulps
    default  : 0 #default SL
end-qos-ulps
```

It is equivalent to the previous example of the shortest policy file, and it is also equivalent to not having policy file at all. Below is an example of simple QoS policy with all the possible keywords:

```
qos-ulps
default                       : 0 # default SL
sdp, port-num 30000           : 0 # SL for application running on
                                  # top of SDP when a destination
                                  # TCP/IPport is 30000
sdp, port-num 10000-20000     : 0
sdp                           : 1 # default SL for any other
                                  # application running on top of SDP
rds                           : 2 # SL for RDS traffic
ipoib, pkey 0x0001            : 0 # SL for IPoIB on partition with
                                  # pkey 0x0001
ipoib                         : 4 # default IPoIB partition,
                                  # pkey=0x7FFF
any, service-id 0x6234        : 6 # match any PR/MPR query with a
                                  # specific Service ID
any, pkey 0x0ABC              : 6 # match any PR/MPR query with a
                                  # specific PKey
srp, target-port-guid 0x1234  : 5 # SRP when SRP Target is located
                                  # on a specified IB port GUID
any, target-port-guid 0x0ABC-0xFFFFF : 6 # match any PR/MPR query
                                  # with a specific target port GUID
end-qos-ulps
```

Similar to the advanced policy definition, matching of PR/MPR queries is done in order of appearance in the QoS policy file such as the first match takes precedence, except for the "default" rule, which is applied only if the query didn't match any other rule. All other sections of the QoS policy file take precedence over the qos-ulps section. That is, if a policy file has both qos-match-rules and qos-ulps sections, then any query is matched first against the rules in the qos-match-rules section, and only if there was no match, the query is matched against the rules in qos-ulps section.

Note that some of these match rules may overlap, so in order to use the simple QoS definition effectively, it is important to understand how each of the ULPs is matched.

9.6.6.1 IPoIB

IPoIB query is matched by PKey or by destination GID, in which case this is the GID of the multicast group that OpenSM creates for each IPoIB partition.

Default PKey for IPoIB partition is 0x7fff, so the following three match rules are equivalent:

```
ipoib              : <SL>
ipoib, pkey 0x7fff : <SL>
any,   pkey 0x7fff : <SL>
```

9.6.6.2 SDP

SDP PR query is matched by Service ID. The Service-ID for SDP is 0x000000000001PPPP, where PPPP are 4 hex digits holding the remote TCP/IP Port Number to connect to. The following two match rules are equivalent:

```
sdp                                                    : <SL>
any, service-id 0x0000000000010000-0x000000000001ffff : <SL>
```

9.6.6.3 RDS

Similar to SDP, RDS PR query is matched by Service ID. The Service ID for RDS is 0x000000000106PPPP, where PPPP are 4 hex digits holding the remote TCP/IP Port Number to connect to. Default port number for RDS is 0x48CA, which makes a default Service-ID 0x00000000010648CA. The following two match rules are equivalent:

```
rds                                : <SL>
any, service-id 0x00000000010648CA : <SL>
```

9.6.6.4 SRP

Service ID for SRP varies from storage vendor to vendor, thus SRP query is matched by the target IB port GUID. The following two match rules are equivalent:

```
srp, target-port-guid 0x1234  : <SL>
any, target-port-guid 0x1234  : <SL>
```

Note that any of the above ULPs might contain target port GUID in the PR query, so in order for these queries not to be recognized by the QoS manager as SRP, the SRP match rule (or any match rule that refers to the target port guid only) should be placed at the end of the qos-ulps match rules.

9.6.6.5 MPI

SL for MPI is manually configured by MPI admin. OpenSM is not forcing any SL on the MPI traffic, and that's why it is the only ULP that did not appear in the qos-ulps section.

9.6.7 SL2VL Mapping and VL Arbitration

OpenSM cached options file has a set of QoS related configuration parameters, that are used to configure SL2VL mapping and VL arbitration on IB ports. These parameters are:

- Max VLs: the maximum number of VLs that will be on the subnet
- High limit: the limit of High Priority component of VL Arbitration table (IBA 7.6.9)
- VLArb low table: Low priority VL Arbitration table (IBA 7.6.9) template
- VLArb high table: High priority VL Arbitration table (IBA 7.6.9) template

- SL2VL: SL2VL Mapping table (IBA 7.6.6) template. It is a list of VLs corresponding to SLs 0-15 (Note that VL15 used here means drop this SL).

There are separate QoS configuration parameters sets for various target types: CAs, routers, switch external ports, and switch's enhanced port 0. The names of such parameters are prefixed by "qos_<type>_" string. Here is a full list of the currently supported sets:

- qos_ca_ - QoS configuration parameters set for CAs.
- qos_rtr_ - parameters set for routers.
- qos_sw0_ - parameters set for switches' port 0.
- qos_swe_ - parameters set for switches' external ports.

Here's the example of typical default values for CAs and switches' external ports (hard-coded in OpenSM initialization):

```
qos_ca_max_vls 15
qos_ca_high_limit 0
qos_ca_vlarb_high 0:4,1:0,2:0,3:0,4:0,5:0,6:0,7:0,8:0,9:0,10:0,11:0,12:0,13:0,14:0
qos_ca_vlarb_low 0:0,1:4,2:4,3:4,4:4,5:4,6:4,7:4,8:4,9:4,10:4,11:4,12:4,13:4,14:4
qos_ca_sl2vl 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,7
qos_swe_max_vls 15
qos_swe_high_limit 0
qos_swe_vlarb_high 0:4,1:0,2:0,3:0,4:0,5:0,6:0,7:0,8:0,9:0,10:0,11:0,12:0,13:0,14:0
qos_swe_vlarb_low 0:0,1:4,2:4,3:4,4:4,5:4,6:4,7:4,8:4,9:4,10:4,11:4,12:4,13:4,14:4
qos_swe_sl2vl 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,7
```

VL arbitration tables (both high and low) are lists of VL/Weight pairs. Each list entry contains a VL number (values from 0-14), and a weighting value (values 0-255), indicating the number of 64 byte units (credits) which may be transmitted from that VL when its turn in the arbitration occurs. A weight of 0 indicates that this entry should be skipped. If a list entry is programmed for VL15 or for a VL that is not supported or is not currently configured by the port, the port may either skip that entry or send from any supported VL for that entry.

Note, that the same VLs may be listed multiple times in the High or Low priority arbitration tables, and, further, it can be listed in both tables. The limit of high-priority VLArb table (qos_<type>_high_limit) indicates the number of high-priority packets that can be transmitted without an opportunity to send a low-priority packet. Specifically, the number of bytes that can be sent is high_limit times 4K bytes.

A high_limit value of 255 indicates that the byte limit is unbounded.

If the 255 value is used, the low priority VLs may be starved.

A value of 0 indicates that only a single packet from the high-priority table may be sent before an opportunity is given to the low-priority table.

Keep in mind that ports usually transmit packets of size equal to MTU. For instance, for 4KB MTU a single packet will require 64 credits, so in order to achieve effective VL arbitration for packets of 4KB MTU, the weighting values for each VL should be multiples of 64.

Below is an example of SL2VL and VL Arbitration configuration on subnet:

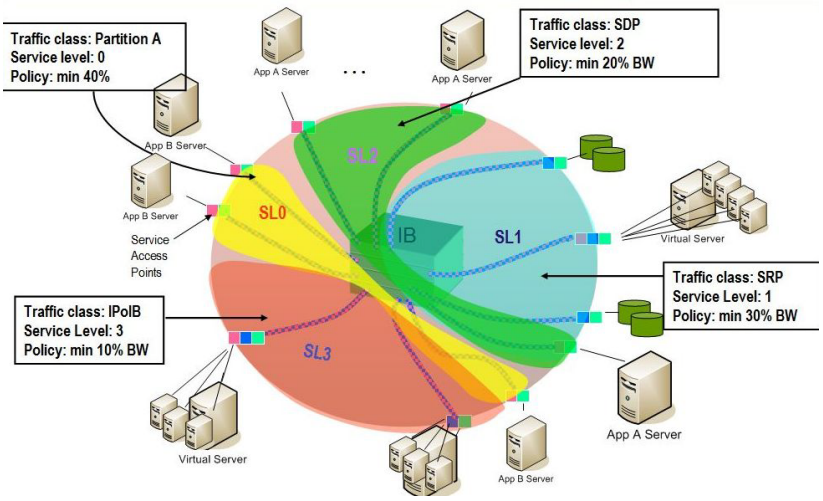
```
qos_ca_max_vls 15
qos_ca_high_limit 6
qos_ca_vlarb_high 0:4
qos_ca_vlarb_low 0:0,1:64,2:128,3:192,4:0,5:64,6:64,7:64
qos_ca_sl2vl 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,7
qos_swe_max_vls 15
qos_swe_high_limit 6
qos_swe_vlarb_high 0:4
qos_swe_vlarb_low 0:0,1:64,2:128,3:192,4:0,5:64,6:64,7:64
qos_swe_sl2vl 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,7
```

In this example, there are 8 VLs configured on subnet: VL0 to VL7. VL0 is defined as a high priority VL, and it is limited to 6 x 4KB = 24KB in a single transmission burst. Such configuration would suilt VL that needs low latency and uses small MTU when transmitting packets. Rest of VLs are defined as low priority VLs with different weights, while VL4 is effectively turned off.

9.6.8 Deployment Example

Figure 4 shows an example of an InfiniBand subnet that has been configured by a QoS manager to provide different service levels for various ULPs.

Figure 4: Example QoS Deployment on InfiniBand Subnet



9.7 QoS Configuration Examples

The following are examples of QoS configuration for different cluster deployments. Each example provides the QoS level assignment and their administration via OpenSM configuration files.

9.7.1 Typical HPC Example: MPI and Lustre

Assignment of QoS Levels

- MPI
  - Separate from I/O load
  - Min BW of 70%
- Storage Control (Lustre MDS)
  - Low latency
- Storage Data (Lustre OST)
  - Min BW 30%

Administration

- MPI is assigned an SL via the command line

```
host1# mpirun -sl 0
```

- OpenSM QoS policy file

In the following policy file example, replace OST* and MDS* with the real port GUIDs.

```
 qos-ulps
     default                                                 :0 # default SL (for
MPI)
     any, target-port-guid OST1,OST2,OST3,OST4:1 # SL for Lustre OST
     any, target-port-guid MDS1,MDS2                         :2 # SL for Lustre
MDS
 end-qos-ulps
```

- OpenSM options file

```
  qos_max_vls 8
  qos_high_limit 0
  qos_vlarb_high 2:1
  qos_vlarb_low 0:96,1:224
  qos_sl2vl 0,1,2,3,4,5,6,7,15,15,15,15,15,15,15,15
```

9.7.2 EDC SOA (2-tier): IPoIB and SRP

The following is an example of QoS configuration for a typical enterprise data center (EDC) with service oriented architecture (SOA), with IPoIB carrying all application traffic and SRP used for storage.

QoS Levels

- Application traffic
  - IPoIB (UD and CM) and SDP
  - Isolated from storage
  - Min BW of 50%
- SRP
  - Min BW 50%
  - Bottleneck at storage nodes

Administration

- OpenSM QoS policy file

In the following policy file example, replace SRPT* with the real SRP Target port GUIDs.

```
qos-ulps
    default                                 :0
    ipoib                                   :1
    sdp                                     :1
    srp, target-port-guid SRPT1,SRPT2,SRPT3 :2
end-qos-ulps
```

- OpenSM options file

```
qos_max_vls 8
 qos_high_limit 0
 qos_vlarb_high 1:32,2:32
 qos_vlarb_low 0:1,
 qos_sl2vl 0,1,2,3,4,5,6,7,15,15,15,15,15,15,15,15
```

9.7.3 EDC (3-tier): IPoIB, RDS, SRP

The following is an example of QoS configuration for an enterprise data center (EDC), with IPoIB carrying all application traffic, RDS for database traffic, and SRP used for storage.

QoS Levels

- Management traffic (ssh)
  - IPoIB management VLAN (partition A)
  - Min BW 10%
- Application traffic
  - IPoIB application VLAN (partition B)
  - Isolated from storage and database

  - Min BW of 30%
- Database Cluster traffic
  - RDS
  - Min BW of 30%
- SRP
  - Min BW 30%
  - Bottleneck at storage nodes

Administration

- OpenSM QoS policy file

In the following policy file example, replace SRPT* with the real SRP Initiator port GUIDs.

```
qos-ulps
    default                                 : 0
    ipoib, pkey 0x8001                      : 1
    ipoib, pkey 0x8002                      : 2
    rds                                     : 3
    srp, target-port-guid SRPT1, SRPT2, SRPT3  : 4
end-qos-ulps
```

- OpenSM options file

```
 qos_max_vls 8
 qos_high_limit 0
 qos_vlarb_high 1:32,2:96,3:96,4:96
 qos_vlarb_low 0:1
 qos_sl2vl 0,1,2,3,4,5,6,7,15,15,15,15,15,15,15,15
```

- Partition configuration file

```
Default=0x7fff,     ipoib : ALL=full;
PartA=0x8001, sl=1, ipoib : ALL=full;
```

9.8 Adaptive Routing

9.8.1 Overview

Adaptive Routing is at beta stage.

Adaptive Routing (AR) enables the switch to select the output port based on the port's load. AR supports two routing modes:

- Free AR: No constraints on output port selection.
- Bounded AR: The switch does not change the output port during the same transmission burst. This mode minimizes the appearance of out-of-order packets.

Adaptive Routing Manager enables and configures Adaptive Routing mechanism on fabric switches. It scans all the fabric switches, deduces which switches support Adaptive Routing and configures the AR functionality on these switches.

Currently, Adaptive Routing Manager supports only link aggregation algorithm. Adaptive Routing Manager configures AR mechanism to allow switches to select output port out of all the ports that are linked to the same remote switch. This algorithm suits any topology with several links between switches. Especially, it suits 3D torus/mesh, where there are several link in each direction of the X/Y/Z axis.

If some switches do not support AR, they will slow down the AR Manager as it may get timeouts on the AR-related queries to these switches.

9.8.2 Installing the Adaptive Routing

Adaptive Routing Manager is a Subnet Manager plug-in, i.e. it is a shared library (libarmgr.so) that is dynamically loaded by the Subnet Manager. Adaptive Routing Manager is installed as a part of Mellanox OFED installation.

9.8.3 Running Subnet Manager with Adaptive Routing Manager

Adaptive Routing (AR) Manager can be enabled/disabled through SM options file.

9.8.3.1 Enabling Adaptive Routing

To enable Adaptive Routing, perform the following:

1. Create the Subnet Manager options file. Run:

```
opensm -c <options-file-name>
```

2. Add '**armgr**' to the '**event_plugin_name**' option in the file:

```
# Event plugin name(s)
event_plugin_name armgr
```

3. Run Subnet Manager with the new options file:

```
opensm -F <options-file-name>
```

Adaptive Routig Manager can read options file with various configuration parameters to fine-tune AR mechanism and AR Manager behavior. Default location of the AR Manager options file is **/etc/opensm/ar_mgr.conf**.

To provide an alternative location, please perform the following:

1. Add '**armgr --conf_file <ar-mgr-options-file-name>**' to the '**event_plugin_options**' option in the file **# Options** string that would be passed to the plugin(s) **event_plugin_options armgr --conf_file <ar-mgr-options-file-name>**
2. Run Subnet Manager with the new options file:

```
opensm -F <options-file-name>
```

See an example of AR Manager options file with all the default values in “Example of Adaptive Routing Manager Options File” on page 153.

9.8.3.2 Disabling Adaptive Routing

There are two ways to disable Adaptive Routing Manager:

1. By disabling it explicitly in the Adaptive Routing configuration file.
2. By removing the 'armgr' option from the Subnet Manager options file.

Adaptive Routing mechanism is automatically disabled once the switch receives setting of the usual linear routing table (LFT).

Therefore, no action is required to clear Adaptive Routing configuration on the switches if you do not wish to use Adaptive Routing.

9.8.4 Querying Adaptive Routing Tables

When Adaptive Routing is active, the content of the usual Linear Forwarding Routing Table on the switch is invalid, thus the standard tools that query LFT (e.g. smpquery, dump_lfts.sh, and others) cannot be used. To query the switch for the content of its Adaptive Routing table, use the 'smparquery' tool that is installed as a part of the Adaptive Routing Manager package. To see its usage details, run 'smparquery -h'.

9.8.5 Adaptive Routing Manager Options File

The default location of the AR Manager options file is /etc/opensm/ar_mgr.conf. To set an alternative location, please perform the following:

1. Add 'armgr --conf_file **<ar-mgr-options-file-name>** to the **event_plugin_option**' option in the file **# Options** string that would be passed to the plugin(s) **event_plugin_options armgr --conf_file <ar-mgr-options-file-name>**
2. Run Subnet Manager with the new options file:

```
opensm -F <options-file-name>'
```

AR Manager options file contains two types of parameters:

1. General options - Options which describe the AR Manager behavior and the AR parameters that will be applied to all the switches in the fabric.
2. Per-switch options - Options which describe specific switch behavior.

Note the following:

- Adaptive Routing configuration file is case sensitive.
- You can specify options for nonexisting switch GUID. These options will be ignored until a switch with a matching GUID will be added to the fabric.
- Adaptive Routing configuration file is parsed every AR Manager cycle, which in turn is executed at every heavy sweep of the Subnet Manager.
- If the AR Manager fails to parse the options file, default settings for all the options will be used.

9.8.5.1 General AR Manager Options

Table 7 - Adaptive Routing Manager Options File

Option File	Description	Values
ENABLE: <true|false>	Enable/disable Adaptive Routing on fabric switches. Note that if a switch was identified by AR Manager as device that does not support AR, AR Manager will not try to enable AR on this switch. If the firmware of this switch was updated to support the AR, the AR Manager will need to be restarted (by restarting Subnet Manager) to allow it to configure the AR on this switch. This option can be changed on-the-fly.	Default: true
AR_MODE: <bounded|free>	Adaptive Routing Mode: • free: no constraints on output port selection • bounded: the switch does not change the output port during the same transmission burst. This mode minimizes the appearance of out-of-order packets This option can be changed on-the-fly.	Default: bounded
AGEING_TIME: <usec>	Applicable to bounded AR mode only. Specifies how much time there should be no traffic in order for the switch to declare a transmission burst as finished and allow changing the output port for the next transmission burst (32-bit value). This option can be changed on-the-fly.	Default: 30
MAX_ERRORS: <N> ERROR_WINDOW: <N>	When number of errors exceeds 'MAX_ERRORS' of send/receive errors or timeouts in less than 'ERROR_WINDOW' seconds, the AR Manager will abort, returning control back to the Subnet Manager. This option can be changed on-the-fly.	Values for both options: [0-0xffff] • MAX_ERRORS = 0: zero tollerance - abort configuration on first error. Default:10 • ERROR_WINDOW = 0: mechanism disabled - no error checking. Default: 5
LOG_FILE: <full path>	AR Manager log file. This option can be changed on-the-fly.	Default: /var/log/armgr.log
LOG_SIZE: <size in MB>	This option defines maximal AR Manager log file size in MB. The logfile will be truncated and restarted upon reaching this limit. This option cannot be changed on-the-fly.	0: unlimited log file size. Default: 5

Per-switch AR Options

A user can provide per-switch configuration options with the following syntax:

```
SWITCH <GUID> {
                <switch option 1>;
                <switch option 2>;
                ...
                }
```

The following are the per-switch options:

Table 8 - Adaptive Routing Manager Pre-Switch Options File

Option File	Description	Values
ENABLE: <true|false>	Allows you to enable/disable the AR on this switch. If the general ENABLE option value is set to 'false', then this per-switch option is ignored. This option can be changed on the fly.	Default: true
AGEING_TIME: <usec>	Applicable to bounded AR mode only. Specifies how much time there should be no traffic in order for the switch to declare a transmission burst as finished and allow changing the output port for the next transmission burst (32-bit value). In the pre-switch options file this option refers to the particular switch only This option can be changed on-the-fly.	Default: 30

Example of Adaptive Routing Manager Options File

```
ENABLE: true;
 LOG_FILE: /tmp/ar_mgr.log;
 LOG_SIZE: 100;
 MAX_ERRORS: 10;
 ERROR_WINDOW: 5;

SWITCH 0x12345 {
ENABLE: true;
AGEING_TIME: 77;
 }

SWITCH 0x0002c902004050f8 {
AGEING_TIME: 44;
 }

SWITCH 0xabcde {
ENABLE: false;
 }
```

9.9 Congestion Control

9.9.1 Congestion Control Overview

Congestion Control Manager is a Subnet Manager (SM) plug-in, i.e. it is a shared library (libcc-mgr.so) that is dynamically loaded by the Subnet Manager. Congestion Control Manager is installed as part of Mellanox OFED installation.

The Congestion Control mechanism controls traffic entry into a network and attempts to avoid oversubscription of any of the processing or link capabilities of the intermediate nodes and networks. Additionally, is takes resource reducing steps by reducing the rate of sending packets. Congestion Control Manager enables and configures Congestion Control mechanism on fabric nodes (HCAs and switches).

9.9.2 Running OpenSM with Congestion Control Manager

Congestion Control (CC) Manager can be enabled/disabled through SM options file. To do so, perform the following:

1. Create the file. Run:

```
opensm -c <options-file-name>'
```

2. Find the '**event_plugin_name**' option in the file, and add '**ccmgr**' to it.

```
# Event plugin name(s)
event_plugin_name ccmgr
```

3. Run the SM with the new options file: '**opensm -F <options-file-name>**'

Once the Congestion Control is enabled on the fabric nodes, to completely disable Congestion Control, you will need to actively turn it off. Running the SM w/o the CC Manager is not sufficient, as the hardware still continues to function in accordance to the previous CC configuration.

For further information on how to turn OFF CC, please refer to Section 9.9.3, “Configuring Congestion Control Manager,” on page 154

9.9.3 Configuring Congestion Control Manager

Congestion Control (CC) Manager comes with a predefined set of setting. However, you can fine-tune the CC mechanism and CC Manager behavior by modifying some of the options. To do so, perform the following:

1. Find the '**event_plugin_options**' option in the SM options file, and add the following:

```
conf_file <cc-mgr-options-file-name>':
# Options string that would be passed to the plugin(s)
event_plugin_options ccmgr --conf_file <cc-mgr-options-file-name>
```

2. Run the SM with the new options file: '**opensm -F <options-file-name>'**

To turn CC OFF, set 'enable' to 'FALSE' in the Congestion Control Manager configuration file, and run OpenSM ones with this configuration.

For the full list of CC Manager options with all the default values, See “Configuring Congestion Control Manager” on page 154.

For further details on the list of CC Manager options, please refer to the IB spec.

9.9.4 Configuring Congestion Control Manager Main Settings

To fine-tune CC mechanism and CC Manager behavior, and set the CC manager main settings, perform the following:

- To enables/disables Congestion Control mechanism on the fabric nodes, set the following parameter:

```
enable
```

  - The values are: **<TRUE | FALSE>**.
  - The default is: **True**

- CC manager configures CC mechanism behavior based on the fabric size. The larger the fabric is, the more aggressive CC mechanism is in its response to congestion. To manually modify CC manager behavior by providing it with an arbitrary fabric size, set the following parameter:

```
num_hosts
```

  - The values are: **[0-48K]**.
  - The default is: **0** (base on the CCT calculation on the current subnet size)

- The smaller the number value of the parameter, the faster HCAs will respond to the congestion and will throttle the traffic. Note that if the number is too low, it will result in suboptimal bandwidth. To change the mean number of packets between marking eligible packets with a FECN, set the following parameter:

```
marking_rate
```

  - The values are: **[0-0xffff]**.
  - The default is: **0xa**

- You can set the minimal packet size that can be marked with FECN. Any packet less than this size [bytes] will not be marked with FECN. To do so, set the following parameter:

```
packet_size
```

  - The values are: **[0-0x3fc0]**.
  - The default is: **0x200**

- When number of errors exceeds 'max_errors' of send/receive errors or timeouts in less than 'error_window' seconds, the CC MGR will abort and will allow OpenSM to proceed. To do so, set the following parameter:

```
max_errors
error_window
```

  - The values are:

    max_errors = 0: zero tollerance - abort configuration on first error

```
error_window = 0: mechanism disabled - no error checking.[0-48K]
```

- The default is: `5`

9.9.4.1 Congestion Control Manager Options File

Table 9 - Congestion Control Manager General Options File

Option File	Description	Values
enable	Enables/disables Congestion Control mechanism on the fabric nodes.	Values: <TRUE | FALSE> Default: True
num_hosts	Indicates the number of nodes. The CC table values are calculated based on this number.	Values: [0-48K] Default: 0 (base on the CCT calculation on the current sub-net size)

Table 10 - Congestion Control Manager Switch Options File

Option File	Description	Values
threshold	Indicates how aggressive the congestion marking should be.	[0-0xf] • 0 - no packet marking, • 0xf - very aggressive Default: 0xf
marking_rate	The mean number of packets between marking eligible packets with a FECN	Values: [0-0xffff] Default: 0xa
packet_size	Any packet less than this size [bytes] will not be marked with FECN.	Values: [0-0x3fc0] Default: 0x200

Table 11 - Congestion Control Manager CA Options File

Option File	Desctiption	Values
port_control	Specifies the Congestion Control attribute for this port	Values: • 0 - QP based congestion control, • 1 - SL/Port based congestion control Default: 0
ca_control_map	An array of sixteen bits, one for each SL. Each bit indicates whether or not the corresponding SL entry is to be modified.	Values: 0xffff
ccti_increase	Sets the CC Table Index (CCTI) increase.	Default: 1
trigger_threshold	Sets the trigger threshold.	Default: 2
ccti_min	Sets the CC Table Index (CCTI) minimum.	Default: 0

Table 11 - Congestion Control Manager CA Options File

Option File	Desctiption	Values
cct	Sets all the CC table entries to a specified value . The first entry will remain 0, whereas last value will be set to the rest of the table.	Values: <comma-separated list> Default: 0 When the value is set to 0, the CCT calculation is based on the number of nodes.
ccti_timer	Sets for all SL's the given ccti timer.	Default: 0 When the value is set to 0, the CCT calculation is based on the number of nodes.

Table 12 - Congestion Control Manager CC MGR Options File

Option File	Desctiption	Values
max_errors error_window	When number of errors exceeds 'max_errors' of send/receive errors or timeouts in less than 'error_window' seconds, the CC MGR will abort and will allow OpenSM to proceed.	Values: • max_errors = 0: zero tollerance - abort configuration on first error. • error_window = 0: mechanism disabled - no error checking. Default: 5
cc_statistics_cycle	Enables CC MGR to collect statistics from all nodes every cc_statistics_cycle [seconds]	Default: 0 When the value is set to 0, no statistics are collected.

10 InfiniBand Fabric Diagnostic Utilities

10.1 Overview

The diagnostic utilities described in this chapter provide means for debugging the connectivity and status of InfiniBand (IB) devices in a fabric. The tools are:

- Section 10.3,“ibdiagnet (of ibutils2) - IB Net Diagnostic,” on page 160
- Section 10.4,“ibdiagnet (of ibutils) - IB Net Diagnostic,” on page 162
- Section 10.5,“ibdiagpath - IB diagnostic path,” on page 165
- Section 10.6,“ibv_devices,” on page 167
- Section 10.7,“ibv_devinfo,” on page 168
- Section 10.8,“ibdev2netdev,” on page 169
- Section 10.9,“ibstatus,” on page 170
- Section 10.10,“ibportstate,” on page 172
- Section 10.11,“ibroute,” on page 175
- Section 10.12,“smpquery,” on page 178
- Section 10.13,“perfquery,” on page 182
- Section 10.14,“ibcheckerrs,” on page 184
- Section 10.15,“mstflint,” on page 186
- Section 10.16,“ibv_asyncwatch,” on page 190
- Section 10.17,“ibdump,” on page 190

10.2 Utilities Usage

This section first describes common configuration, interface, and addressing for all the tools in the package. Then it provides detailed descriptions of the tools themselves including: operation, synopsis and options descriptions, error codes, and examples.

10.2.1 Common Configuration, Interface and Addressing

Topology File (Optional)

An InfiniBand fabric is composed of switches and channel adapter (HCA/TCA) devices. To identify devices in a fabric (or even in one switch system), each device is given a GUID (a MAC equivalent). Since a GUID is a non-user-friendly string of characters, it is better to alias it to a meaningful, user-given name. For this objective, the IB Diagnostic Tools can be provided with a “topology file”, which is an optional configuration file specifying the IB fabric topology in user-given names.

For diagnostic tools to fully support the topology file, the user may need to provide the local system name (if the local hostname is not used in the topology file).

To specify a topology file to a diagnostic tool use one of the following two options:

1. On the command line, specify the file name using the option ‘`-t <topology file name>`’
2. Define the environment variable IBDIAG_TOPO_FILE

To specify the local system name to an diagnostic tool use one of the following two options:

1. On the command line, specify the system name using the option '`-s <local system name>`'
2. Define the environment variable IBDIAG_SYS_NAME

10.2.2 IB Interface Definition

The diagnostic tools installed on a machine connect to the IB fabric by means of an HCA port through which they send MADs. To specify this port to an IB diagnostic tool use one of the following options:

1. On the command line, specify the port number using the option '-p <local port number>' (see below)
2. Define the environment variable IBDIAG_PORT_NUM

In case more than one HCA device is installed on the local machine, it is necessary to specify the device's index to the tool as well. For this use on of the following options:

1. On the command line, specify the index of the local device using the following option: '-i <index of local device>'
2. Define the environment variable IBDIAG_DEV_IDX

10.2.3 Addressing

This section applies to the ibdiagpath tool only. A tool command may require defining the destination device or port to which it applies.

The following addressing modes can be used to define the IB ports:

- Using a Directed Route to the destination: (Tool option '-d')

  This option defines a directed route of output port numbers from the local port to the destination.

- Using port LIDs: (Tool option '-l'):

  In this mode, the source and destination ports are defined by means of their LIDs. If the fabric is configured to allow multiple LIDs per port, then using any of them is valid for defining a port.

- Using port names defined in the topology file: (Tool option '-n')

  This option refers to the source and destination ports by the names defined in the topology file. (Therefore, this option is relevant only if a topology file is specified to the tool.) In this mode, the tool uses the names to extract the port LIDs from the matched topology, then the tool operates as in the '-l' option.

10.3 ibdiagnet (of ibutils2) - IB Net Diagnostic

This version of ibdiagnet is included in the ibutils2 package, and it is run by default after installing Mellanox OFED. To use this ibdiagnet version, run: ibdiagnet

Please see ibutils2_release_notes.txt for additional information and known issues.

ibdiagnet scans the fabric using directed route packets and extracts all the available information regarding its connectivity and devices. It then produces the following files in the output directory (which is defined by the -o option described below).

10.3.1 SYNOPSYS

```
ibdiagnet [-i <dev-name>] [-p <port-num>]
        [-pm] [-pc] [-P <<PM>=<Value>>]
        [-r] [-u]
        [-lw <1x|4x|8x|12x>] [-ls <2.5|5|10>]
        [-skip <ibdiag stage>]
        [-o <out-dir>] [-h] [-V]
```

OPTIONS:

```
-i|--device <dev-name>
                          Specify the name of the device of the port used
                         to connect to the IB fabric (in case of
                         multiple devices on the local system)
-p|--port <port-num>
                         Specify the local device's port number used to
                         connect to the IB fabric
-pm Dump all pm          Counters values into ibdiagnet.pm
-pc                      Reset all the fabric links pmCounters
-P|--counter <<PM>=<Value>>
                         Print any provided pm that is greater than its
                         provided value
-r|--routing             Provide a report of the fabric qualities
-u|--fat_tree            Indicate that UpDown credit loop checking
                         should be done against automatically
                         determined roots
-lw <1x|4x|8x|12x>       Specify the expected link width
-ls <2.5|5|10>           Specify the expected link speed
-skip <ibdiag check>
                         Skip the execution of the given stage.
                         Applicable to the following stages:
                         dup_guids|lids|links|sm|nodes_info|all
                         (default = None)
-o|--output_path <out-dir>
                         Specify the directory where the output files
                         will be placed
--screen_num_errs        Specify the threshold for printing errors to
                         screen (default = 5).
                         Placed (default = /var/tmp/ibdiagnet2/)
-h|--help                Print this help message
-V|--version             Print the version of the tool
```

10.3.2 Output Files

Table 13 lists the ibdiagnet output files that are placed under /var/tmp/ibdiagnet2.

Table 13 - ibdiagnet (of ibutils2) Output Files

Output File	Description
ibdiagnet2.lst	Fabric links in LST format
ibdiagnet2.sm	Subnet Manager
ibdiagnet2.pm	Ports Counters
ibdiagnet2.fdbs	Unicast FDBs
ibdiagnet2.mcfdbs	Multicast FDBx
ibdiagnet2.nodes_info	Information on nodes
ibdiagnet2.db_csv	ibdiagnet internal database

An ibdiagnet run performs the following stages:

- Fabric discovery
- Duplicated GUIDs detection
- Links in INIT state and unresponsive links detection
- Counters fetch
- Error counters check
- Routing checks
- Link width and speed checks

10.3.3 Return Codes

```
0 - Success
1 - Failure (with description)
```

10.4 ibdiagnet (of ibutils) - IB Net Diagnostic

This version of ibdiagnet is included in the ibutils package, and it is not run by default after installing Mellanox OFED. To use this ibdiagnet version and not that of the ibutils package, you need to specify the full path: /opt/bin/ibdiagnet

ibdiagnet scans the fabric using directed route packets and extracts all the available information regarding its connectivity and devices. It then produces the following files in the output directory (which is defined by the -o option described below).

10.4.1 SYNOPSYS

```
bdiagnet [-c <count>] [-v] [-r] [-o <out-dir>] [-t <topo-file>]
          [-s <sys-name>] [-i <dev-index>] [-p <port-num>] [-wt]
          [-pm] [-pc] [-P <<PM>=<Value>>] [-lw <1x|4x|12x>] [-ls
          <2.5|5|10>]
          [-skip <ibdiag_check/s>] [-load_db <db_file>]
```

OPTIONS:

```
-c <count>        Min number of packets to be sent across each link (default
                = 10)
-v                Enable verbose mode
-r                Provides a report of the fabric qualities
-t <topo-file>    Specifies the topology file name
-s <sys-name>     Specifies the local system name. Meaningful only if a
                topology file is specified
-i <dev-index>    Specifies the index of the device of the port used to
                connect to the IB fabric (in case of multiple devices on
                the local system)
-p <port-num>     Specifies the local device's port num used to connect to
                the IB fabric
-o <out-dir>      Specifies the directory where the output files will be
                placed (default = /tmp)
-lw <1x|4x|12x>   Specifies the expected link width
-ls <2.5|5|10>    Specifies the expected link speed
-pm               Dump all the fabric links, pm Counters into ibdiagnet.pm
-pc               Reset all the fabric links pmCounters
-P <PM=<Trash>>   If any of the provided pm is greater then its provided
                value, print it to screen
-skip <skip-option(s)>Skip the executions of the selected checks. Skip
                options (one or more can be specified): dup_guids
                zero_guids pm logical_state part ipoib all
-wt <file-name>   Write out the discovered topology into the given file.
                This flag is useful if you later want to check for
                changes from the current state of the fabric. A directory
                named ibdiag_ibnl is also created by this option, and
                holds the IBNL files required to load this topology.
                To use these files you will need to set the environment
                variable named IBDM_IBNL_PATH to that directory.
                The directory is located in /tmp or in the output
                directory provided by the -o flag.
-load_db <file-name>>Load subnet data from the given .db file, and skip
                subnet discovery stage.
                  Note: Some of the checks require actual subnet discovery,
                and therefore would not run when load_db is specified.
                These checks are: Duplicated/zero guids, link state, SMs
                status.
-h|--help         Prints the help page information
-V|--version      Prints the version of the tool
--vars            Prints the tool's environment variables and their values
```

10.4.2 Output Files

Table 14 - ibdiagnet (of ibutils) Output Files

Output File	Description
ibdiagnet.log	A dump of all the application reports generate according to the provided flags
ibdiagnet.lst	List of all the nodes, ports and links in the fabric
ibdiagnet.fdbs	A dump of the unicast forwarding tables of the fabric switches
ibdiag-net.mcfdbs	A dump of the multicast forwarding tables of the fabric switches
ibdiag-net.masks	In case of duplicate port/node Guids, these file include the map between masked Guid and real Guids
ibdiagnet.sm	List of all the SM (state and priority) in the fabric
ibdiagnet.pm	A dump of the pm Counters values, of the fabric links
ibdiagnet.pkey	A dump of the the existing partitions and their member host ports
ibdiagnet.mcg	A dump of the multicast groups, their properties and member host ports
ibdiagnet.db	A dump of the internal subnet database. This file can be loaded in later runs using the -load_db option

In addition to generating the files above, the discovery phase also checks for duplicate node/port GUIDs in the IB fabric. If such an error is detected, it is displayed on the standard output. After the discovery phase is completed, directed route packets are sent multiple times (according to the -c option) to detect possible problematic paths on which packets may be lost. Such paths are explored, and a report of the suspected bad links is displayed on the standard output.

After scanning the fabric, if the `-r` option is provided, a full report of the fabric qualities is displayed. This report includes:

- SM report
- Number of nodes and systems
- Hop-count information: maximal hop-count, an example path, and a hop-count histogram
- All CA-to-CA paths traced
- Credit loop report
- mgid-mlid-HCAs multicast group and report
- Partitions report
- IPoIB report

In case the IB fabric includes only one CA, then CA-to-CA paths are not reported. Furthermore, if a topology file is provided, ibdiagnet uses the names defined in it for the output reports.

10.4.3 ERROR CODES

```
1 - Failed to fully discover the fabric
2 - Failed to parse command line options
3 - Failed to intract with IB fabric
4 - Failed to use local device or local port
5 - Failed to use Topology File
6 - Failed to load requierd Package
```

10.5 ibdiagpath - IB diagnostic path

`ibdiagpath` traces a path between two end-points and provides information regarding the nodes and ports traversed along the path. It utilizes device specific health queries for the different devices along the path.

The way `ibdiagpath` operates depends on the addressing mode used on the command line. If directed route addressing is used (-d flag), the local node is the source node and the route to the destination port is known apriori. On the other hand, if LID-route (or by-name) addressing is employed, then the source and destination ports of a route are specified by their LIDs (or by the names defined in the topology file). In this case, the actual path from the local port to the source port, and from the source port to the destination port, is defined by means of Subnet Management Linear Forwarding Table queries of the switch nodes along that path. Therefore, the path cannot be predicted as it may change.

`ibdiagpath` should not be supplied with contradicting local ports by the -p and -d flags (see synopsis descriptions below). In other words, when ibdiagpath is provided with the options -p and -d together, the first port in the direct route must be equal to the one specified in the “-p” option. Otherwise, an error is reported.

When ibdiagpath queries for the performance counters along the path between the source and destination ports, it always traverses the LID route, even if a directed route is specified. If along the LID route one or more links are not in the ACTIVE state, ibdiagpath reports an error.

Moreover, the tool allows omitting the source node in LID-route addressing, in which case the local port on the machine running the tool is assumed to be the source.

10.5.1 SYNOPSYS

```
ibdiagpath {-n <[src-name,]dst-name>|-l <[src-lid,]dst-lid>|-d
            <p1,p2,p3,...>} [-c <count>] [-v] [-t <topo-file>]
            [-s <sys-name>] [-ic<dev-index>]c[-p <port-num>]
            [-o <out-dir>][-lw <1x|4x|12x>] [-ls <2.5|5|10>][-pm] [-pc]
            [-P <<PM counter>=<Trash Limit>>]
```

OPTIONS:

```
-n <[src-name,]dst-name>
                          Names of the source and destination ports (as
                        defined in the topology file; source may be omit
                        ted -> local port is assumed to be the source)
-l <[src-lid,]dst-lid>
                          Source and destination LIDs (source may be omit
                        ted --> the local port is assumed to be the
                        source)
-d <p1,p2,p3,...>        Directed route from the local node (which is the
                        source) and the destination node
-c <count>                The minimal number of packets to be sent across
                        each link (default = 100)
-v                        Enable verbose mode
-t <topo-file>            Specifies the topology file name
-s <sys-name>              Specifies the local system name. Meaningful only
                        if a topology file is specified
-i <dev-index>            Specifies the index of the device of the port
                        used to connect to the IB fabric (in case of
                        multiple devices on the local system)
-p <port-num>              Specifies the local device's port number used to
                        connect to the IB fabric
-o <out-dir>              Specifies the directory where the output files
                        will be placed (default = /tmp)
-lw <1x|4x|12x>           Specifies the expected link width
-ls <2.5|5|10>            Specifies the expected link speed
-pm                       Dump all the fabric links, pm Counters into
                        ibdiagnet.pm
-pc                       Reset all the fabric links pmCounters
-P <PM=<Trash>>           If any of the provided pm is greater then its
                        provided value, print it to screen
-h|--help                 Prints the help page information
-V|--version              Prints the version of the tool
--vars                    Prints the tool's environment variables and
                        their values
```

10.5.2 Output Files

Table 15 - ibdiagpath Output Files

Output File	Description
ibdiagpath.log	A dump of all the application reports generated according to the provided flags
ibdiagnet.pm	A dump of the Performance Counters values, of the fabric links

10.5.3 ERROR CODES

```
1 - The path traced is un-healthy
2 - Failed to parse command line options
3 - More then 64 hops are required for traversing the local port to the
    "Source" port and then to the "Destination" port
4 - Unable to traverse the LFT data from source to destination
5 - Failed to use Topology File
6 - Failed to load required Package
```

10.6 ibv_devices

Applicable Hardware

All InfiniBand devices.

Description

Lists InfiniBand devices available for use from userspace, including node GUIDs.

Synopsis

```
ibv_devices
```

Examples

1. List the names of all available InfiniBand devices.

```
> ibv_devices
    device                 node GUID
    ------              ----------------
    mthca0              0002c9000101d150
    mlx4_0              0000000000073895
```

10.7 ibv_devinfo

Applicable Hardware

All InfiniBand devices.

Description

Queries InfiniBand devices and prints about them information that is available for use from user-space.

Synopsis

```
ibv_devinfo [-d <device>] [-i <port>] [-l] [-v]
```

Table 16 lists the various flags of the command.

Table 16 - ibv_devinfo Flags and Options

Flag	Optional / Mandatory	Default (If Not Specified)	Description
-d <device> --ib-dev=<device>	Optional	First found device	Run the command for the provided IB device 'device'
-i <port> --ib-port=<port>	Optional	All device ports	Query the specified device port <port>
-l --list	Optional	Inactive	Only list the names of InfiniBand devices
-v --verbose	Optional	Inactive	Print all available information about the InfiniBand device(s)

Examples

1. List the names of all available InfiniBand devices.

```
> ibv_devinfo -l
2 HCAs found:
        mthca0
        mlx4_0
```

2. Query the device mlx4_0 and print user-available information for its Port 2.

```
> ibv_devinfo -d mlx4_0 -i 2
hca_id: mlx4_0
        fw_ver:                         2.5.944
        node_guid:                      0000:0000:0007:3895
```

```
        sys_image_guid:                 0000:0000:0007:3898
        vendor_id:                      0x02c9
        vendor_part_id:                 25418
        hw_ver:                         0xA0
        board_id:                       MT_04A0140005
        phys_port_cnt:                  2
                port:   2
                        state:                  PORT_ACTIVE (4)
                        max_mtu:                2048 (4)
                        active_mtu:             2048 (4)
                        sm_lid:                 1
                        port_lid:               1
                        port_lmc:               0x00
```

10.8 ibdev2netdev

ibdev2netdev enables association between IB devices and ports and the associated net device. Additionally it reports the state of the net device link.

10.8.1 SYNOPSYS

```
 ibdiagnet [-v] [-h]
```

OPTIONS:

```
 -v Enable verbose mode. Adds additional information such as: Device ID, Part Number,
Card Name, Firmware version, IB port state.
 -h Print help messages.
```

Example:

```
 sw417:~/BXOFED-1.5.2-20101128-1524 # ibdev2netdev -v
 mlx4_0 (MT26428 - MT1006X00034) FALCON QDR      fw 2.7.9288 port 1 (ACTIVE) ==> eth5
(Down)
 mlx4_0 (MT26428 - MT1006X00034) FALCON QDR      fw 2.7.9288 port 1 (ACTIVE) ==> ib0
(Down)
 mlx4_0 (MT26428 - MT1006X00034) FALCON QDR      fw 2.7.9288 port 2 (DOWN  ) ==> ib1
(Down)
 mlx4_1 (MT26448 - MT1023X00777) Hawk Dual Port  fw 2.7.9400 port 1 (DOWN  ) ==> eth2
(Down)
 mlx4_1 (MT26448 - MT1023X00777) Hawk Dual Port  fw 2.7.9400 port 2 (DOWN  ) ==> eth3
(Down)
 sw417:~/BXOFED-1.5.2-20101128-1524 # ibdev2netdev
 mlx4_0 port 1 ==> eth5 (Down)
 mlx4_0 port 1 ==> ib0 (Down)
 mlx4_0 port 2 ==> ib1 (Down)
 mlx4_1 port 1 ==> eth2 (Down)
```

```
mlx4_1 port 2 ==> eth3 (Down)
```

10.9 ibstatus

Applicable Hardware

All InfiniBand devices.

Description

Displays basic information obtained from the local InfiniBand driver. Output includes LID, SMLID, port state, port physical state, port width and port rate.

Synopsis

```
ibstatus  [-h] [<device name>[:<port>]]*
```

Table 17 lists the various flags of the command.

Table 17 - ibstatus Flags and Options

Flag	Optional / Mandatory	Default (If Not Specified)	Description
-h	Optional		Print the help menu
<device>	Optional	All devices	Print information for the specified device. May specify more than one device
<port>	Optional, but requires specifying a device name	All ports of the specified device	Print information for the specified port only (of the specified device)

Examples

1. List the status of all available InfiniBand devices and their ports.

```
> ibstatus
Infiniband device 'mlx4_0' port 1 status:
        default gid:     fe80:0000:0000:0000:0000:0000:0007:3896
        base lid:        0x3
        sm lid:          0x3
        state:           4: ACTIVE
        phys state:      5: LinkUp
        rate:            20 Gb/sec (4X DDR)

Infiniband device 'mlx4_0' port 2 status:
        default gid:     fe80:0000:0000:0000:0000:0000:0007:3897
        base lid:        0x1
```

```
        sm lid:          0x1
        state:           4: ACTIVE
        phys state:      5: LinkUp
        rate:            20 Gb/sec (4X DDR)

Infiniband device 'mthca0' port 1 status:
        default gid:     fe80:0000:0000:0000:0002:c900:0101:d151
        base lid:        0x0
        sm lid:          0x0
        state:           2: INIT
        phys state:      5: LinkUp
        rate:            10 Gb/sec (4X)

Infiniband device 'mthca0' port 2 status:
        default gid:     fe80:0000:0000:0000:0002:c900:0101:d152
        base lid:        0x0
        sm lid:          0x0
        state:           2: INIT
        phys state:      5: LinkUp
        rate:            10 Gb/sec (4X)
```

2. List the status of specific ports of specific devices.

```
> ibstatus mthca0:1 mlx4_0:2
Infiniband device 'mthca0' port 1 status:
        default gid:     fe80:0000:0000:0000:0002:c900:0101:d151
        base lid:        0x0
        sm lid:          0x0
        state:           2: INIT
        phys state:      5: LinkUp
        rate:            10 Gb/sec (4X)

Infiniband device 'mlx4_0' port 2 status:
        default gid:     fe80:0000:0000:0000:0000:0000:0007:3897
        base lid:        0x1
        sm lid:          0x1
        state:           4: ACTIVE
        phys state:      5: LinkUp
        rate:            20 Gb/sec (4X DDR)
```

10.10 ibportstate

Applicable Hardware

All InfiniBand devices.

Description

Enables querying the logical (link) and physical port states of an InfiniBand port. It also allows adjusting the link speed that is enabled on any InfiniBand port.

If the queried port is a *swich* port, then ibportstate can be used to

- disable, enable or reset the port
- validate the port's link width and speed against the peer port

Synopsis

```
 ibportstate [-d] [-e] [-v] [-V] [-D] [-G] [-s <smlid>] \           [-C <ca_name>] [-P
<ca_port>] [-t <timeout_ms>]       \                          [<dest dr_path|lid|guid>]
<portnum> [<op> [<value>]]
```

Table 18 lists the various flags of the command.

Table 18 - ibportstate Flags and Options

Flag	Optional / Mandatory	Default (If Not Specified)	Description
-h(help)	Optional		Print the help menu
-d(ebug)	Optional		Raise the IB debug level. May be used several times for higher debug levels (-ddd or -d -d -d)
-e(rr_show)	Optional		Show send and receive errors (timeouts and others)
-v(erbose)	Optional		Increase verbosity level. May be used several times for additional verbosity (-vvv or -v -v -v)
-V(ersion)	Optional		Show version info
-D(irect)	Optional		Use directed path address arguments. The path is a comma separated list of out ports. Examples: '0' – self port '0,1,2,1,4' – out via port 1, then 2, ...
-G(uid)	Optional		Use GUID address argument. In most cases, it is the Port GUID. Example: '0x08f1040023'
-s <smlid>	Optional		Use <smlid> as the target lid for SM/SA queries
-C <ca_name>	Optional		Use the specified channel adapter or router
-P <ca_port>	Optional		Use the specified port

Table 18 - ibportstate Flags and Options (Continued)

Flag	Optional / Mandatory	Default (If Not Specified)	Description
-t <timeout_ms>	Optional		Override the default timeout for the solicited MADs [msec]
<dest dr_path | lid | guid>	Optional		Destination's directed path, LID, or GUID.
<portnum>	Optional		Destination's port number
<op> [<value>]	Optional	query	Define the allowed port operations: enable, disable, reset, speed, and query

In case of multiple channel adapters (CAs) or multiple ports without a CA/port being specified, a port is chosen by the utility according to the following criteria:

1. The first ACTIVE port that is found.
2. If not found, the first port that is UP (physical link state is LinkUp).

Examples

1. Query the status of Port 1 of CA mlx4_0 (using ibstatus) and use its output (the LID – 3 in this case) to obtain additional link information using ibportstate.

```
>  ibstatus mlx4_0:1
Infiniband device 'mlx4_0' port 1 status:
        default gid:     fe80:0000:0000:0000:0000:0000:9289:3895
        base lid:        0x3
        sm lid:          0x3
        state:           2: INIT
        phys state:      5: LinkUp
        rate:            20 Gb/sec (4X DDR)

> ibportstate -C mlx4_0 3 1 query
PortInfo:
# Port info: Lid 3 port 1
LinkState:.....................Initialize
PhysLinkState:.................LinkUp
LinkWidthSupported:............1X or 4X
LinkWidthEnabled:..............1X or 4X
LinkWidthActive:...............4X
LinkSpeedSupported:............2.5 Gbps or 5.0 Gbps
LinkSpeedEnabled:..............2.5 Gbps or 5.0 Gbps
LinkSpeedActive:...............5.0 Gbps
```

2. Query the status of two channel adapters using directed paths.

```
>  ibportstate -C mlx4_0 -D 0 1
PortInfo:
# Port info: DR path slid 65535; dlid 65535; 0 port 1
LinkState:.....................Initialize
PhysLinkState:.................LinkUp
LinkWidthSupported:............1X or 4X
LinkWidthEnabled:..............1X or 4X
LinkWidthActive:...............4X
LinkSpeedSupported:............2.5 Gbps or 5.0 Gbps
LinkSpeedEnabled:..............2.5 Gbps or 5.0 Gbps
LinkSpeedActive:...............5.0 Gbps

>  ibportstate -C mthca0 -D 0 1
PortInfo:
# Port info: DR path slid 65535; dlid 65535; 0 port 1
LinkState:.....................Down
PhysLinkState:.................Polling
LinkWidthSupported:............1X or 4X
LinkWidthEnabled:..............1X or 4X
LinkWidthActive:...............4X
LinkSpeedSupported:............2.5 Gbps
LinkSpeedEnabled:..............2.5 Gbps
LinkSpeedActive:...............2.5 Gbps
```

3. Change the speed of a port.

```
# First query for current configuration
>  ibportstate -C mlx4_0 -D 0 1
PortInfo:
# Port info: DR path slid 65535; dlid 65535; 0 port 1
LinkState:.....................Initialize
PhysLinkState:.................LinkUp
LinkWidthSupported:............1X or 4X
LinkWidthEnabled:..............1X or 4X
LinkWidthActive:...............4X
LinkSpeedSupported:............2.5 Gbps or 5.0 Gbps
LinkSpeedEnabled:..............2.5 Gbps or 5.0 Gbps
LinkSpeedActive:...............5.0 Gbps

# Now change the enabled link speed
>  ibportstate -C mlx4_0 -D 0 1 speed 2
ibportstate -C mlx4_0 -D 0 1 speed 2
Initial PortInfo:
# Port info: DR path slid 65535; dlid 65535; 0 port 1
LinkSpeedEnabled:..............2.5 Gbps
```

```
After PortInfo set:
# Port info: DR path slid 65535; dlid 65535; 0 port 1
LinkSpeedEnabled:.............5.0 Gbps (IBA extension)

# Show the new configuration
>  ibportstate -C mlx4_0 -D 0 1
PortInfo:
# Port info: DR path slid 65535; dlid 65535; 0 port 1
LinkState:.....................Initialize
PhysLinkState:.................LinkUp
LinkWidthSupported:............1X or 4X
LinkWidthEnabled:..............1X or 4X
LinkWidthActive:...............4X
LinkSpeedSupported:............2.5 Gbps or 5.0 Gbps
LinkSpeedEnabled:..............5.0 Gbps (IBA extension)
LinkSpeedActive:...............5.0 Gbps
```

10.11 ibroute

Applicable Hardware

InfiniBand switches.

Description

Uses SMPs to display the forwarding tables—unicast (LinearForwardingTable or LFT) or multicast (MulticastForwardingTable or MFT)—for the specified switch LID and the optional lid (mlid) range. The default range is all valid entries in the range 1 to FDBTop.

Synopsis

```
 ibroute [-h] [-d] [-v] [-V] [-a] [-n] [-D] [-G] [-M] [-s <smlid>] \[-C <ca_name>] [-P
<ca_port>] [       -t <timeout_ms>] \              [<dest dr_path|lid|guid> [<star-
tlid> [<endlid>]]]
```

Table 19 lists the various flags of the command.

Table 19 - ibportstate Flags and Options

Flag	Optional / Mandatory	Default (If Not Specified)	Description
-h(help)	Optional		Print the help menu
-d(ebug)	Optional		Raise the IB debug level. May be used several times for higher debug levels (-ddd or -d -d -d)
-a(ll)	Optional		Show all LIDs in range, including invalid entries

Table 19 - ibportstate Flags and Options

Flag	Optional / Mandatory	Default (If Not Specified)	Description
-v(erbose)	Optional		Increase verbosity level. May be used several times for additional verbosity (-vvv or -v -v -v)
-V(ersion)	Optional		Show version info
-a(ll)	Optional		Show all LIDs in range, including invalid entries
-n(o_dests)	Optional		Do not try to resolve destinations
-D(irect)	Optional		Use directed path address arguments. The path is a comma separated list of out ports. Examples: '0' – self port '0,1,2,1,4' – out via port 1, then 2, ...
-G(uid)	Optional		Use GUID address argument. In most cases, it is the Port GUID. Example: '0x08f1040023'
-M(ulticast)	Optional		Show multicast forwarding tables. The parameters <startlid> and <endlid> specify the MLID range.
-s <smlid>	Optional		Use <smlid> as the target LID for SM/SA queries
-C <ca_name>	Optional		Use the specified channel adapter or router
-P <ca_port>	Optional		Use the specified port
-t <timeout_ms>	Optional		Override the default timeout for the solicited MADs [msec]
<dest dr_path | lid | guid>	Optional		Destination's directed path, LID, or GUID
<startlid>	Optional		Starting LID in an MLID range
<endlid>	Optional		Ending LID in an MLID range

Examples

1. Dump all Lids with valid out ports of the switch with Lid 2.

```
 > ibroute 2
 Unicast lids [0x0-0x8] of switch Lid 2 guid 0x0002c902fffff00a (MT47396 Infiniscale-III
Mellanox Technologies):
   Lid  Out   Destination
        Port     Info
 0x0002 000 : (Switch portguid 0x0002c902fffff00a: 'MT47396 Infiniscale-III Mellanox
Technologies')
 0x0003 021 : (Switch portguid 0x000b8cffff004016: 'MT47396 Infiniscale-III Mellanox
Technologies')
```

```
 0x0006 007 : (Channel Adapter portguid 0x0002c90300001039: 'sw137 HCA-1')
 0x0007 021 : (Channel Adapter portguid 0x0002c9020025874a: 'sw157 HCA-1')
 0x0008 008 : (Channel Adapter portguid 0x0002c902002582cd: 'sw136 HCA-1')
 5 valid lids dumped
```

2. Dump all Lids with valid out ports of the switch with Lid 2.

```
 > ibroute 2
 Unicast lids [0x0-0x8] of switch Lid 2 guid 0x0002c902fffff00a (MT47396 Infiniscale-III
Mellanox Technologies):
   Lid  Out   Destination
        Port     Info
 0x0002 000 : (Switch portguid 0x0002c902fffff00a: 'MT47396 Infiniscale-III Mellanox
Technologies')
 0x0003 021 : (Switch portguid 0x000b8cffff004016: 'MT47396 Infiniscale-III Mellanox
Technologies')
 0x0006 007 : (Channel Adapter portguid 0x0002c90300001039: 'sw137 HCA-1')
 0x0007 021 : (Channel Adapter portguid 0x0002c9020025874a: 'sw157 HCA-1')
 0x0008 008 : (Channel Adapter portguid 0x0002c902002582cd: 'sw136 HCA-1')
 5 valid lids dumped
```

3. Dump all Lids in the range 3 to 7 with valid out ports of the switch with Lid 2.

```
 > ibroute 2 3 7
 Unicast lids [0x3-0x7] of switch Lid 2 guid 0x0002c902fffff00a (MT47396 Infiniscale-III
Mellanox Technologies):
   Lid  Out   Destination
        Port     Info
 0x0003 021 : (Switch portguid 0x000b8cffff004016: 'MT47396 Infiniscale-III Mellanox
Technologies')
 0x0006 007 : (Channel Adapter portguid 0x0002c90300001039: 'sw137 HCA-1')
 0x0007 021 : (Channel Adapter portguid 0x0002c9020025874a: 'sw157 HCA-1')
 3 valid lids dumped
```

4. Dump all Lids with valid out ports of the switch with portguid `0x000b8cffff004016`.

```
 > ibroute -G 0x000b8cffff004016
 Unicast lids [0x0-0x8] of switch Lid 3 guid 0x000b8cffff004016 (MT47396 Infiniscale-III
Mellanox Technologies):
   Lid  Out   Destination
        Port     Info
 0x0002 023 : (Switch portguid 0x0002c902fffff00a: 'MT47396 Infiniscale-III Mellanox
Technologies')
 0x0003 000 : (Switch portguid 0x000b8cffff004016: 'MT47396 Infiniscale-III Mellanox
Technologies')
 0x0006 023 : (Channel Adapter portguid 0x0002c90300001039: 'sw137 HCA-1')
 0x0007 020 : (Channel Adapter portguid 0x0002c9020025874a: 'sw157 HCA-1')
 0x0008 024 : (Channel Adapter portguid 0x0002c902002582cd: 'sw136 HCA-1')
```

```
 5 valid lids dumped
```

5. Dump all non-empty mlids of switch with Lid 3.

```
 > ibroute -M 3
 Multicast mlids [0xc000-0xc3ff] of switch Lid 3 guid 0x000b8cffff004016 (MT47396
Infiniscale-III Mellanox Technologies):
            0                   1                   2
     Ports: 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4
  MLid
 0xc000                                             x
 0xc001                                             x
 0xc002                                             x
 0xc003                                             x
 0xc020                 x
 0xc021                 x
 0xc022                 x
 0xc023                 x
 0xc024                 x
 0xc040                 x
 0xc041                 x
 0xc042                 x
 12 valid mlids dumped
```

10.12 smpquery

Applicable Hardware

All InfiniBand devices.

Description

Provides a basic subset of standard SMP queries to query Subnet management attributes such as node info, node description, switch info, and port info.

Synopsys

```
 smpquery [-h] [-d] [-e] [-v] [-D] [-G] [-s <smlid>] [-V]         [-C <ca_name>] [-P
<ca_port>] [-t <timeout_ms>]        [--node-name-map <node-name-map>]        <op> <dest
dr_path|lid|guid> [op params]
```

Table 20 lists the various flags of the command.

Table 20 - smpquery Flags and Options

Flag	Optional / Mandatory	Default (If Not Specified)	Description
-h(help)	Optional		Print the help menu

Table 20 - smpquery Flags and Options

Flag	Optional / Mandatory	Default (If Not Specified)	Description
-d(ebug)	Optional		Raise the IB debug level. May be used several times for higher debug levels (-ddd or -d -d -d)
-e(rr_show)	Optional		Show send and receive errors (timeouts and others)
-v(erbose)	Optional		Increase verbosity level. May be used several times for additional verbosity (-vvv or -v -v -v)
-D(irect)	Optional		Use directed path address arguments. The path is a comma separated list of out ports. Examples: ‘0’ – self port ‘0,1,2,1,4’ – out via port 1, then 2, ...
-G(uid)	Optional		Use GUID address argument. In most cases, it is the Port GUID. Example: ‘0x08f1040023’
-s <smlid>	Optional		Use <smlid> as the target LID for SM/SA queries
-V(ersion)	Optional		Show version info
-C <ca_name>	Optional		Use the specified channel adapter or router
-P <ca_port>	Optional		Use the specified port
-t <timeout_ms>	Optional		Override the default timeout for the solicited MADs [msec]
<op>	Mandatory		Supported operations: nodeinfo <addr> nodedesc <addr> portinfo <addr> [<portnum>] switchinfo <addr> pkeys <addr> [<portnum>] sl2vl <addr> [<portnum>] vlarb <addr> [<portnum>] guids <addr>
<dest dr_path | lid | guid>	Optional		Destination’s directed path, LID, or GUID

Examples

1. Query PortInfo by LID, with port modifier.

```
> smpquery portinfo 1 1
# Port info: Lid 1 port 1
Mkey:.........................0x0000000000000000
GidPrefix:....................0xfe80000000000000
Lid:..........................0x0001
```

```
SMLid:.........................0x0001
CapMask:.......................0x251086a
                               IsSM
                               IsTrapSupported
                               IsAutomaticMigrationSupported
                               IsSLMappingSupported
                               IsSystemImageGUIDsupported
                               IsCommunicatonManagementSupported
                               IsVendorClassSupported
                               IsCapabilityMaskNoticeSupported
                               IsClientRegistrationSupported
DiagCode:......................0x0000
MkeyLeasePeriod:...............0
LocalPort:.....................1
LinkWidthEnabled:..............1X or 4X
LinkWidthSupported:............1X or 4X
LinkWidthActive:...............4X
LinkSpeedSupported:............2.5 Gbps or 5.0 Gbps
LinkState:.....................Active
PhysLinkState:.................LinkUp
LinkDownDefState:..............Polling
ProtectBits:...................0
LMC:...........................0
LinkSpeedActive:...............5.0 Gbps
LinkSpeedEnabled:..............2.5 Gbps or 5.0 Gbps
NeighborMTU:...................2048
SMSL:..........................0
VLCap:.........................VL0-7
InitType:......................0x00
VLHighLimit:...................4
VLArbHighCap:..................8
VLArbLowCap:...................8
InitReply:.....................0x00
MtuCap:........................2048
VLStallCount:..................0
HoqLife:.......................31
OperVLs:.......................VL0-3
PartEnforceInb:................0
PartEnforceOutb:...............0
FilterRawInb:..................0
FilterRawOutb:.................0
MkeyViolations:................0
PkeyViolations:................0
QkeyViolations:................0
GuidCap:.......................128
ClientReregister:..............0
```

```
SubnetTimeout:.................18
RespTimeVal:...................16
LocalPhysErr:..................8
OverrunErr:....................8
MaxCreditHint:.................0
RoundTrip:.....................0
```

2. Query SwitchInfo by GUID.

```
> smpquery -G switchinfo 0x000b8cffff004016
# Switch info: Lid 3
LinearFdbCap:..................49152
RandomFdbCap:..................0
McastFdbCap:...................1024
LinearFdbTop:..................8
DefPort:.......................0
DefMcastPrimPort:..............0
DefMcastNotPrimPort:...........0
LifeTime:......................18
StateChange:...................0
LidsPerPort:...................0
PartEnforceCap:................32
InboundPartEnf:................1
OutboundPartEnf:...............1
FilterRawInbound:..............1
FilterRawOutbound:.............1
EnhancedPort0:.................0
```

3. Query NodeInfo by direct route.

```
> smpquery -D nodeinfo 0
# Node info: DR path slid 65535; dlid 65535; 0
BaseVers:......................1
ClassVers:.....................1
NodeType:......................Channel Adapter
NumPorts:......................2
SystemGuid:....................0x0002c9030000103b
Guid:..........................0x0002c90300001038
PortGuid:......................0x0002c90300001039
PartCap:.......................128
DevId:.........................0x634a
Revision:......................0x000000a0
LocalPort:.....................1
VendorId:......................0x0002c9
```

10.13 perfquery

Applicable Hardware

All InfiniBand devices.

Description

Queries InfiniBand ports' performance and error counters. Optionally, it displays aggregated counters for all ports of a node. It can also reset counters after reading them or simply reset them.

Synopsys

```
 perfquery [-h] [-d] [-G] [-a] [-l] [-r] [-C <ca_name>] [-P <ca_port>] [-R]         [-t
<timeout_ms>] [-V] [<lid|guid> [[port][reset_mask]]]
```

Table 21 lists the various flags of the command.

Table 21 - perfquery Flags and Options

Flag	Optional / Mandatory	Default (If Not Specified)	Description
-h(help)	Optional		Print the help menu
-d(ebug)	Optional		Raise the IB debug level. May be used several times for higher debug levels (-ddd or -d -d -d)
-G(uid)	Optional		Use GUID address argument. In most cases, it is the Port GUID. Example: '0x08f1040023'
-a	Optional		Apply query to all ports
-l	Optional		Loop ports
-r	Optional		Reset the counters after reading them
-C <ca_name>	Optional		Use the specified channel adapter or router
-P <ca_port>	Optional		Use the specified port
-R	Optional		Reset the counters
-t <timeout_ms>	Optional		Override the default timeout for the solicited MADs [msec]
-V(ersion)	Optional		Show version info
<lid | guid> [[port][reset_ mask]]	Optional		LID or GUID

Examples

```
perfquery -r 32 1       # read performance counters and reset
perfquery -e -r 32 1    # read extended performance counters and reset
perfquery -R 0x20 1     # reset performance counters of port 1 only
perfquery -e -R 0x20 1  # reset extended performance counters of port 1 only
perfquery -R -a 32      # reset performance counters of all ports
perfquery -R 32 2 0x0fff        # reset only error counters of port 2
perfquery -R 32 2 0xf000        # reset only non-error counters of port 2
```

1. Read local port's performance counters.

```
> perfquery
# Port counters: Lid 6 port 1
PortSelect:......................1
CounterSelect:...................0x1000
SymbolErrors:....................0
LinkRecovers:....................0
LinkDowned:......................0
RcvErrors:.......................0
RcvRemotePhysErrors:.............0
RcvSwRelayErrors:................0
XmtDiscards:.....................0
XmtConstraintErrors:.............0
RcvConstraintErrors:.............0
LinkIntegrityErrors:.............0
ExcBufOverrunErrors:.............0
VL15Dropped:.....................0
XmtData:.........................55178210
RcvData:.........................55174680
XmtPkts:.........................766366
RcvPkts:.........................766315
```

2. Read performance counters from LID 2, all ports.

```
> smpquery -a 2
# Port counters: Lid 2 port 255
PortSelect:......................255
CounterSelect:...................0x0100
SymbolErrors:....................65535
LinkRecovers:....................255
LinkDowned:......................16
RcvErrors:.......................657
RcvRemotePhysErrors:.............0
RcvSwRelayErrors:................70
XmtDiscards:.....................488
```

```
XmtConstraintErrors:.....0
RcvConstraintErrors:.....0
LinkIntegrityErrors:.....0
ExcBufOverrunErrors:.....0
VL15Dropped:.....0
XmtData:.....129840354
RcvData:.....129529906
XmtPkts:.....1803332
RcvPkts:.....1799018
```

3. Read then reset performance counters from LID 2, port 1.

```
> perfquery  -r 2 1
# Port counters: Lid 2 port 1
PortSelect:.....1
CounterSelect:.....0x0100
SymbolErrors:.....0
LinkRecovers:.....0
LinkDowned:.....0
RcvErrors:.....0
RcvRemotePhysErrors:.....0
RcvSwRelayErrors:.....0
XmtDiscards:.....3
XmtConstraintErrors:.....0
RcvConstraintErrors:.....0
LinkIntegrityErrors:.....0
ExcBufOverrunErrors:.....0
VL15Dropped:.....0
XmtData:.....0
RcvData:.....0
XmtPkts:.....0
RcvPkts:.....0
```

10.14 ibcheckerrs

Applicable Hardware

All InfiniBand devices.

Description

Validates an IB port (or node) and reports errors in counters above threshold.

Check specified port (or node) and report errors that surpassed their predefined threshold. Port address is lid unless -G option is used to specify a GUID address. The predefined thresholds can be dumped using the -s option, and a user defined threshold_file (using the same format as the dump) can be specified using the -t <file> option.

Synopsis

```
 ibcheckerrs [-h] [-b] [-v] [-G] [-T <threshold_file>] [-s] [-N |-nocolor] [-C ca_name]
[-P ca_port] [-t timeout_ms] <lid|guid>  [<port>]
```

Table 22 lists the various flags of the command.

Table 22 - ibcheckerrs Flags and Options

Flag	Optional / Mandatory	Default (If Not Specified)	Description
-h(help)	Optional		Print the help menu
-b	Optional		Print in brief mode. Reduce the output to show only if errors are present, not what they are
-v(erbose)	Optional		Increase verbosity level. May be used several times for additional verbosity (-vvv or -v -v -v)
-G(uid)	Optional		Use GUID address argument. In most cases, it is the Port GUID. Example: '0x08f1040023'
-T <threshold_file>	Optional		Use specified threshold file
-s	Optional		Show the predefined thresholds
-N | -nocolor	Optional	color mode	Use mono mode rather than color mode
-C <ca_name>	Optional		Use the specified channel adapter or router
-P <ca_port>	Optional		Use the specified port
-t <timeout_ms>	Optional		Override the default timeout for the solicited MADs [msec]
<lid | guid>	Mandatory with -G flag		Use the specified port's or node's LID/GUID (with -G option)
[<port>]	Mandatory without -G flag		Use the specified port

Examples

1. Check aggregated node counter for LID 0x2.

```
> ibcheckerrs 2
#warn: counter SymbolErrors = 65535     (threshold 10) lid 2 port 255
#warn: counter LinkRecovers = 255       (threshold 10) lid 2 port 255
#warn: counter LinkDowned = 12  (threshold 10) lid 2 port 255
#warn: counter RcvErrors = 565  (threshold 10) lid 2 port 255
#warn: counter XmtDiscards = 441        (threshold 100) lid 2 port 255
```

```
Error check on lid 2 (MT47396 Infiniscale-III Mellanox Technologies) port all:  FAILED
```

2. Check port counters for LID 2 Port 1.

```
> ibcheckerrs -v 2 1
Error check on lid 2 (MT47396 Infiniscale-III Mellanox Technologies) port 1:  OK
```

3. Check the LID2 Port 1 using the specified threshold file.

```
> cat thresh1
SymbolErrors=10
LinkRecovers=10
LinkDowned=10
RcvErrors=10
RcvRemotePhysErrors=100
RcvSwRelayErrors=100
XmtDiscards=100
XmtConstraintErrors=100
RcvConstraintErrors=100
LinkIntegrityErrors=10
ExcBufOverrunErrors=10
VL15Dropped=100
> ibcheckerrs -v -T thresh1 2 1
Error check on lid 2 (MT47396 Infiniscale-III Mellanox Technologies) port 1:  OK
```

10.15 mstflint

Applicable Hardware

Mellanox InfiniBand and Ethernet devices and network adapter cards.

Description

Queries and burns a binary firmware-image file on non-volatile (Flash) memories of Mellanox InfiniBand and Ethernet network adapters. The tool requires root privileges for Flash access.

If you purchased a standard Mellanox Technologies network adapter card, please download the firmware image from www.mellanox.com > Downloads > Firmware. If you purchased a non-standard card from a vendor other than Mellanox Technologies, please contact your vendor.

To run mstflint, you must know the device location on the PCI bus. See Example 1 for details.

Synopsis

```
mstflint [switches...] <command> [parameters...]
```

Table 23 lists the various switches of the utility, and Table 24 lists its commands.

Table 23 - mstflint Switches (Sheet 1 of 2)

Switch	Affected/ Relevant Commands	Description
-h		Print the help menu
-hh		Print an extended help menu
-d[evice] <device>	All	Specify the device to which the Flash is connected.
-guid <GUID>	burn, sg	GUID base value. 4 GUIDs are automatically assigned to the following values: guid -> node GUID guid+1 -> port1 guid+2 -> port2 guid+3 -> system image GUID. Note: Port2 guid will be assigned even for a single port HCA; the HCA ignores this value.
-guids <GUIDs...>	burn, sg	4 GUIDs must be specified here. The specified GUIDs are assigned the following values, repectively: node, port1, port2 and system image GUID. Note: Port2 guid must be specified even for a single port HCA; the HCA ignores this value. It can be set to 0x0.
-mac <MAC>	burn, sg	MAC address base value. Two MACs are automatically assigned to the following values: mac -> port1 mac+1 -> port2 Note: This switch is applicable only for Mellanox Technologies Ethernet products.
-macs <MACs...>	burn, sg	Two MACs must be specified here. The specified MACs are assigned to port1 and port2, repectively. Note: This switch is applicable only for Mellanox Technologies Ethernet products.
-blank_guids	burn	Burn the image with blank GUIDs and MACs (where applicable). These values can be set later using the sg command – see Table 24 below.
-clear_semaphore	No commands allowed	Force clear the Flash semaphore on the device. No command is allowed when this switch is used. Warning: May result in system instability or Flash corruption if the device or another application is currently using the Flash.
-i[mage] <image>	burn, verify	Binary image file
-qq	burn, query	Run a quick query. When specified, mstflint will not perform full image integrity checks during the query operation. This may shorten execution time when running over slow interfaces (e.g., I2C, MTUSB-1).

Table 23 - mstflint Switches (Sheet 2 of 2)

Switch	Affected/ Relevant Commands	Description
-nofs	burn	Burn image in a non-failsafe manner
-skip_is	burn	Allow burning the firmware image without updating the invariant sector. This is to ensure failsafe burning even when an invariant sector difference is detected.
-byte_mode	burn, write	Shift address when accessing Flash internal registers. May be required for burn/write commands when accessing certain Flash types.
-s[ilent]	burn	Do not print burn progress messages
-y[es]	All	Non-interactive mode: Assume the answer is “yes” to all questions.
-no	All	Non-interactive mode: Assume the answer is “no” to all questions.
-vsd <string>	burn	Write this string of up to 208 characters to VSD upon a burn command.
-use_image_ps	burn	Burn vsd as it appears in the given image - do not keep existing VSD on Flash.
-dual_image	burn	Make the burn process burn two images on Flash. The current default failsafe burn process burns a single image (in alternating locations).
-v		Print version info

Table 24 - mstflint Commands (Sheet 1 of 2)

Command	Description
b[urn]	Burn Flash
q[uery]	Query miscellaneous Flash/firmware characteristics
v[erify]	Verify the entire Flash
bb	Burn Block: Burn the given image as is, without running any checks
sg	Set GUIDs
ri <out-file>	Read the firmware image on the Flash into the specified file
dc <out-file>	Dump Configuration: Print a firmware configuration file for the given image to the specified output file
e[rase] <addr>	Erase sector
rw <addr>	Read one DWORD from Flash
ww <addr> < data>	Write one DWORD to Flash

Table 24 - mstflint Commands (Sheet 2 of 2)

Command	Description
wwne <addr>	Write one DWORD to Flash without sector erase
wbne <addr> <size> <data...>	Write a data block to Flash without sector erase
rb <addr> <size> [out-file]	Read a data block from Flash
swreset	SW reset the target InfniScale IV device. This command is supported only in the In-Band access method.

Possible command return values are:

```
 0 - successful completion
 1 - error has occurred
 7 - the burn command was aborted because firmware is current
```

Examples

1. Find Mellanox Technologies's ConnectX® VPI cards with PCI Express running at 2.5GT/s and InfiniBand ports at DDR / or Ethernet ports at 10GigE.

```
 > /sbin/lspci -d 15b3:634a
 04:00.0 InfiniBand: Mellanox Technologies MT25418 [ConnectX IB DDR, PCIe 2.0 2.5GT/s]
(rev a0).
```

   In the example above, 15b3 is Mellanox Technologies's vendor number (in hexadecimal), and 634a is the device's PCI Device ID (in hexadecimal). The number string 04:00.0 identifies the device in the form bus:dev.fn.

The PCI Device IDs of Mellanox Technologies' devices can be obtained from the PCI ID Repository Website at http://pci-ids.ucw.cz/read/PC/15b3.

2. Verify the ConnectX firmware using its ID (using the results of the example above).

```
 > mstflint -d 04:00.0 v
      ConnectX failsafe image. Start address: 80000. Chunk size 80000:

 NOTE: The addresses below are contiguous logical addresses. Physical addresses on flash
may be different, based on the image start address and chunk size

      /0x00000038-0x000010db (0x0010a4)/ (BOOT2) - OK
      /0x000010dc-0x00004947 (0x00386c)/ (BOOT2) - OK
      /0x00004948-0x000052c7 (0x000980)/ (Configuration) - OK
      /0x000052c8-0x0000530b (0x000044)/ (GUID) - OK
```

```
      /0x0000530c-0x0000542f (0x000124)/ (Image Info) - OK
      /0x00005430-0x0000634f (0x000f20)/ (DDR) - OK
      /0x00006350-0x0000f29b (0x008f4c)/ (DDR) - OK
      /0x0000f29c-0x0004749b (0x038200)/ (DDR) - OK
      /0x0004749c-0x0005913f (0x011ca4)/ (DDR) - OK
      /0x00059140-0x0007a123 (0x020fe4)/ (DDR) - OK
      /0x0007a124-0x0007bdff (0x001cdc)/ (DDR) - OK
      /0x0007be00-0x0007eb97 (0x002d98)/ (DDR) - OK
      /0x0007eb98-0x0007f0af (0x000518)/ (Configuration) - OK
      /0x0007f0b0-0x0007f0fb (0x00004c)/ (Jump addresses) - OK
      /0x0007f0fc-0x0007f2a7 (0x0001ac)/ (FW Configuration) - OK

FW image verification succeeded. Image is bootable.
```

10.16 ibv_asyncwatch

Applicable Hardware

All InfiniBand devices.

Description

Display asynchronous events forwarded to userspace for an InfiniBand device.

Synopsis

```
ibv_asyncwatch
```

Examples

1. Display asynchronous events.

```
> ibv_asyncwatch
mlx4_0: async event FD 4
```

10.17 ibdump

Applicable Hardware

Mellanox ConnectX® / ConnectX®-2 / ConnectX®-3 adapter devices.

Description

Dump InfiniBand traffic that flows to and from Mellanox Technologies ConnectX family adapters InfiniBand ports. The dump file can be loaded by the Wireshark tool for graphical traffic analysis.

The following describes a work flow for local HCA (adapter) sniffing:

- Run ibdump with the desired options

- Run the application that you wish its traffic to be analyzed
- Stop ibdump (CTRL-c) or wait for the data buffer to fill (in --mem-mode)
- Open Wireshark and load the generated file

How to Get Wireshark:

Download the current release from www.wireshark.org for a Linux or Windows environment. See the ibdump_release_notes.txt file for more details.

Although ibdump is a Linux application, the generated .pcap file may be analyzed on either operating system.

Synopsis

```
ibdump [options]
```

Table 25 lists the various flags of the command.

Table 25 - ibdump Options

Flag	Optional / Mandatory	Default (If Not Specified)	Description
-h(help)	Optional		Print the help menu
-d, --ib-dev=<dev>	Optional	First device found	Use IB device <dev>
-i, --ib-port=<port>	Optional	1	Use port <port> of IB device
-o, --output=<file>	Optional	sniffer.pcap	Dump file name
-b, --max-burst=<log2 burst>	Optional	12 - 4096 entries	log2 of the maximal burst size that can be captured with no packet loss. Each entry takes ~ MTU bytes of memory
--mem-mode <size>	Optional		When specified, packets are written to the dump file only after the capture is stopped. It is faster than the default mode (less chance for packet loss), but it uses more memory. In this mode, ibdump stops after <size> bytes are captured.
--decap	Optional		Decapsulate port mirroring headers. Should be used when capturing RSPAN traffic.

Examples

1. Run ibdump.

```
> ibdump
```

```
 ------------------------------------------------
 IB device                     : "mlx4_0"
 IB port                       : 1
 Dump file                     : sniffer.pcap
 Sniffer WQEs (max burst size)  : 4096
 ------------------------------------------------

Initiating resources ...
searching for IB devices in host
Port active_mtu=2048
MR was registered with addr=0x60d850, lkey=0x28042601, rkey=0x28042601, flags=0x1
QP was created, QP number=0x4004a

Ready to capture (Press ^c to stop):
```

Appendix A: Mellanox FlexBoot

A.1 Overview

Mellanox FlexBoot is a multiprotocol remote boot technology. FlexBoot supports remote Boot over InfiniBand (BoIB) and over Ethernet.

Using Mellanox Virtual Protocol Interconnect (VPI) technologies available in ConnectX® adapters, FlexBoot gives IT Managers' the choice to boot from a remote storage target (iSCSI target) or a LAN target (Ethernet Remote Boot Server) using a single ROM image on Mellanox ConnectX products.

FlexBoot is based on the open source project iPXE available at http://www.ipxe.org.

FlexBoot first initializes the adapter device, senses the port protocol – Ethernet or InfiniBand, and brings up the port. Then it connects to a DHCP server to obtain its assigned IP address and network parameters, and also to obtain the source location of the kernel/OS to boot from. The DHCP server instructs FlexBoot to access the kernel/OS through a TFTP server, an iSCSI target, or some other service.

For an InfiniBand port, Mellanox FlexBoot implements a network driver with IP over IB acting as the transport layer. IP over IB is part of the *Mellanox OFED for Linux* software package (see www.mellanox.com > Products > InfiniBand/VPI SW/Drivers).

The binary code is exported by the device as an expansion ROM image.

A.1.1 Supported Mellanox Adapter Devices and Firmware

The package supports all ConnectX® / ConnectX®-2 / ConnectX®-3 network adapter devices and cards. Specifically, adapter products responding to the following PCI Device IDs are supported:

ConnectX / ConnectX-2 / ConnectX-3 devices:

- Decimal 25408 (Hexadecimal: 6340)
- Decimal 25418 (Hexadecimal: 634a)
- Decimal 25448 (Hexadecimal: 6368)
- Decimal 26418 (Hexadecimal: 6732)
- Decimal 26428 (Hexadecimal: 673c)
- Decimal 26438 (Hexadecimal: 6746)
- Decimal 26448 (Hexadecimal: 6750)
- Decimal 25458 (Hexadecimal: 6372)
- Decimal 26458 (Hexadecimal: 675a)
- Decimal 26468 (Hexadecimal: 6764)
- Decimal 26478 (Hexadecimal: 676e)
- Decimal 4099 (Hexadecimal: 1003)

A.1.2 Tested Platforms

See the Mellanox FlexBoot Release Notes (`FlexBoot_release_notes.txt`).

A.1.3 FlexBoot in Mellanox OFED

The FlexBoot package is provided as a tarball (.tgz extension) containing the files specified in Appendix A.1.1, “Supported Mellanox Adapter Devices and Firmware” (page 193)..

1. A PXE ROM image file for each of the supported Mellanox network adapter devices. Specifically, the following images are included:

   ConnectX / ConnectX-2 / ConnectX-3 images:

   - **`ConnectX_FlexBoot_<PCI Device ID>_ROM-<version>.mrom`**

     where the number after the "**`ConnectX_FlexBoot_`**" prefix indicates the corresponding PCI Device ID of the ConnectX / ConnectX-2 / ConnectX-3 device.

2. Additional documents under **`docs/dhcp`**:

A.2 Burning the Expansion ROM Image

A.2.1 Burning the Image on ConnectX® / ConnectX®-2 / ConnectX®-3

This section is valid for ConnectX® / ConnectX®-2 devices with firmware versions 2.8.0600 or later and ConnectX®-3 firmware.

Prerequisites

1. Expansion ROM Image

   The expansion ROM images are provided as part of the Mellanox FlexBoot package and are listed in the release notes file `FlexBoot_release_notes.txt`.

2. Firmware Burning Tools

   You need to install the Mellanox Firmware Tools (MFT) package (version 2.7.0 or later) in order to burn the PXE ROM image. To download MFT, see Firmware Tools under www.mellanox.com > Downloads.

Image Burning Procedure

To burn the composite image, perform the following steps:

1. Obtain the MST device name. Run:

```
# mst start
# mst status
```

   The device name will be of the form: `mt<dev_id>_pci{_cr0|conf0}`.[1]

2. Create and burn the composite image. Run:

```
flint -dev <mst device name> brom <expansion ROM image>
```

   Example on Linux:

1. Depending on the OS, the device name may be superceded with a prefix.

```
flint -dev /dev/mst/mt26428_pci_cr0 brom ConnectX_26428_ROM-X_X_XXX.mrom
```

Example on Windows:

```
flint -dev mt26428_pci_cr0 brom ConnectX_26428_ROM-X_X_XXX.mrom
```

Removing the Expansion ROM Image

Remove the expansion ROM image. Run:

```
flint -dev <mst device name> drom
```

When removing the expansion ROM image, you also remove Flexboot from the boot device list.

A.3 Preparing the DHCP Server in Linux Environment

The DHCP server plays a major role in the boot process by assigning IP addresses for FlexBoot clients and instructing the clients where to boot from. FlexBoot requires that the DHCP server run on a machine which supports IP over IB.

A.3.1 Installing the DHCP Server

To add IPoIB support in DHCP (client/server), please refer to **docs/dhcp/README.**

A.3.2 Configuring the DHCP Server

A.3.2.1 For ConnectX Family Devices

When a FlexBoot client boots, it sends the DHCP server various information, including its DHCP client identifier. This identifier is used to distinguish between the various DHCP sessions. The value of the client identifier is composed of a prefix — ff:00:00:00:00:00:02:00:00:02:c9:00 — and an 8-byte port GUID (all separated by colons and represented in hexadecimal digits).

Extracting the Port GUID – Method I

To obtain the port GUID, run the following commands:

The following MFT commands assume that the Mellanox Firmware Tools (MFT) package has been installed on the client machine.

```
host1# mst start
host1# mst status
```

The device name will be of the form: /dev/mst/mt<dev_id>_pci{_cr0|conf0}. Use this device name to obtain the Port GUID via the following query command:

```
flint -d <MST_DEVICE_NAME> q
```

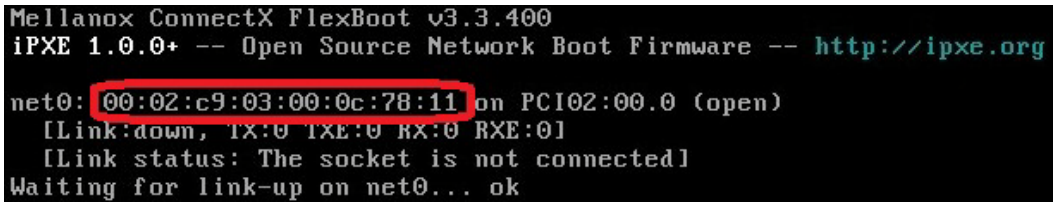
Example with ConnectX-2 QDR (MHJH29B-XTR Dual 4X IB QDR Port, PCIe Gen2 x8, Tall Bracket, RoHS-R6 HCA Card, CX4 Connectors) as the adapter device:

```
Image type:      ConnectX
FW Version:      2.9.1000
Rom Info:        type=PXE  version=3.3.400 devid=26428 proto=VPI
Device ID:       26428
Description:     Node             Port1            Port2            Sys image
GUIDs:           0002c9030005cffa 0002c9030005cffb 0002c9030005cffc
0002c9030005cffd
MACs:            0002c905cffa     0002c905cffb
Board ID:       (MT_0DD0110009)
VSD:
PSID:            MT_0DD0110009
```

Assuming that FlexBoot is connected via Port 1, then the Port GUID is 00:02:c9:03:00:05:cf:fb.

Extracting the Port GUID – Method II

An alternative method for obtaining the port GUID involves booting the client machine via FlexBoot. This requires having a Subnet Manager running on one of the machines in the InfiniBand subnet. The 8 bytes can be captured from the boot session as shown in the figure below.


Mellanox ConnectX FlexBoot v3.3.400
iPXE 1.0.0+ -- Open Source Network Boot Firmware -- http://ipxe.org
net0: 00:02:c9:03:00:0c:78:11 on PCI02:00.0 (open)
[Link:down, TX:0 TXE:0 RX:0 RXE:0]
[Link status: The socket is not connected]
Waiting for link-up on net0... ok

Placing Client Identifiers in /etc/dhcpd.conf

The following is an excerpt of a `/etc/dhcpd.conf` example file showing the format of representing a client machine for the DHCP server.

```
host host1 {
   next-server 11.4.3.7;
   filename "pxelinux.0";
   fixed-address 11.4.3.130;
   option dhcp-client-identifier =
      ff:00:00:00:00:00:02:00:00:02:c9:00:00:02:c9:03:00:00:10:39;
}
```

A.4 Subnet Manager – OpenSM

This section applies to ports configured as InfiniBand only.

FlexBoot requires a Subnet Manager to be running on one of the machines in the IB network. OpenSM is part of the *Mellanox OFED for Linux* software package and can be used to accomplish this. Note that OpenSM may be run on the same host running the DHCP server but it is not mandatory. For details on OpenSM, see "OpenSM – Subnet Manager" on page 108.

To use OpenSM caching for large InfiniBand clusters (> 100 nodes), it is recommended to use the OpenSM options described in Section 9.2.1, "opensm Syntax," on page 108.

A.5 TFTP Server

When you set the 'filename' parameter in your DHCP configuration file to a non-empty filename, the client will ask for this file to be passed through TFTP. For this reason you need to install a TFTP server.

A.6 BIOS Configuration

The expansion ROM image presents itself to the BIOS as a boot device. As a result, the BIOS will add to the list of boot devices "MLNX FlexBoot <ver>" for a ConnectX device. The priority of this list can be modified through BIOS setup.

A.7 Operation

A.7.1 Prerequisites

- Make sure that your client is connected to the server(s)
- The FlexBoot image is already programmed on the adapter card – see Section A.2
- For InfiniBand ports only: Start the Subnet Manager as described in Section A.4
- The DHCP server should be configured and started (see Section 4.6.3.1, "IPoIB Configuration Based on DHCP," on page 79
- Configure and start at least one of the services iSCSI Target (see Section A.10) and/or TFTP (see Section A.5)

A.7.2 Starting Boot

Boot the client machine and enter BIOS setup to configure “MLNX FlexBoot” to be the first on the boot device priority list – see Section A.6.

On dual-port network adapters, the client first attempts to boot from Port 1. If this fails, it switches to boot from Port 2. Note also that the driver waits up to 90 seconds for each port to come up.

If MLNX FlexBoot/iPXE was selected through BIOS setup, the client will boot from FlexBoot. The client will display FlexBoot attributes, sense the port protocol – Ethernet or InfiniBand. In case of an InfiniBand port, the client will also wait for port configuration by the Subnet Manager.

In case sensing the port protocol fails, the port will be configured as an InfiniBand port.

For ConnectX:

```
Mellanox ConnectX FlexBoot v3.3.400
iPXE 1.0.0+ -- Open Source Network Boot Firmware -- http://ipxe.org

net0: 00:02:c9:03:00:0c:78:11 on PCI02:00.0 (open)
  [Link:down, TX:0 TXE:0 RX:0 RXE:0]
  [Link status: The socket is not connected]
Waiting for link-up on net0... ok
```

After configuring the IB/ETH port, the client attempts connecting to the DHCP server to obtain an IP address and the source location of the kernel/OS to boot from.

For ConnectX (InfiniBand):

```
Mellanox ConnectX FlexBoot v3.3.400
iPXE 1.0.0+ -- Open Source Network Boot Firmware -- http://ipxe.org

net0: 00:02:c9:03:00:0c:78:11 on PCI02:00.0 (open)
  [Link:down, TX:0 TXE:0 RX:0 RXE:0]
  [Link status: The socket is not connected]
Waiting for link-up on net0... ok
DHCP (net0 02:02:c9:0c:78:11).... ok
net0: 11.3.12.2/255.255.255.0
Next server: 11.3.12.121
Filename: pxeilinux.0
Root path: /tftpboot/
tftp://11.3.12.121/pxeilinux.0...
```

Next, FlexBoot attempts to boot as directed by the DHCP server.

A.8 Command Line Interface (CLI)

A.8.1 Invoking the CLI

When the boot process begins, the computer starts its Power On Self Test (POST) sequence. Shortly after completion of the POST, the user will be prompted to press CTRL-B to invoke Mellanox FlexBoot CLI. The user has few seconds to press CTRL-B before the message disappears (see figure).

Alternatively, you may skip invoking CLI right after POST and invoke it, instead, right after FlexBoot *starts* booting.

Once the CLI is invoked, you will see the following prompt:

```
iPXE>
```

A.8.2 Operation

The CLI resembles a Linux shell, where the user can run commands to configure and manage one or more PXE port network interfaces. Each port is assigned a network interface called **net*i***, where ***i*** is 0, 1, 2,...<#of interface>. Some commands are general and are applied to all network interfaces. Other commands are port specific, therefore the relevant network interface is specified in the command.

A.8.3 Command Reference

A.8.3.1 ifstat

Displays the available network interfaces (in a similar manner to Linux's ifconfig).

```
iPXE> ifstat
net0: 00:02:c9:03:00:0c:78:11 on PCI02:00.0 (closed)
  [Link:down, TX:8 TXE:2 RX:11 RXE:11]
  [Link status: The socket is not connected]
  [TXE: 2 x "No such file or directory"]
  [RXE: 3 x "The socket is not connected"]
  [RXE: 8 x "Operation canceled"]
net1: 00:02:c9:0c:78:12 on PCI02:00.0 (open)
  [Link:up, TX:12 TXE:0 RX:0 RXE:0]
iPXE> _
```

A.8.3.2 ifopen

Opens the network interface net<x>. The list of network interfaces is available via the ifstat command.

Example:

```
iPXE> ifopen net1
```

A.8.3.3 ifclose

Closes the network interface net<x>. The list of network interfaces is available via the ifstat command.

Example:

```
iPXE> ifclose net1
```

A.8.3.4 autoboot

Starts the boot process from the device(s).

A.8.3.5 sanboot

Starts the boot process of an iSCSI target.

Example:

```
iPXE> sanboot iscsi:11.4.3.7::::iqn.2007-08.7.3.4.11:iscsiboot
```

A.8.3.6 echo

Echoes an environment variable.

Example:

```
iPXE> echo ${root-path}
```

A.8.3.7 dhcp

A network interface attempts to open the network interface and then tries to connect to and communicate with the DHCP server to obtain the IP address and filepath from which the boot will occur.

Example:

```
iPXE> dhcp net1
```

A.8.3.8 help

Displays the available list of commands.

A.8.3.9 exit

Exits from the command line interface.

A.9 Diskless Machines

Mellanox FlexBoot supports booting diskless machines. To enable using an IB/ETH driver, the `initrd` image must include a device driver module and be configured to load that driver.

This can be achieved by adding the device driver module into the `initrd` image and loading it.

The **'initrd'** image of some Linux distributions such as SuSE Linux Enterprise Server and Red Hat Enterprise Linux, cannot be edited prior or during the installation process.

If you need to install Linux distributions over Flexboot, please replace your **'initrd'** images with the images found at:

www.mellanox.com > Products > Adapter IB/VPI SW > FlexBoot (Download Tab).

A.9.1 Case I: InfiniBand Ports

The IB driver requires loading the following modules in the specified order (see Section A.9.1.1 for an example):

- ib_addr.ko
- ib_core.ko
- ib_mad.ko
- ib_sa.ko
- ib_cm.ko
- ib_uverbs.ko
- ib_ucm.ko
- ib_umad.ko
- iw_cm.ko
- rdma_cm.ko
- rdma_ucm.ko
- mlx4_core.ko
- mlx4_ib.ko
- ib_mthca.ko
- ipoib_helper.ko – this module is *not* required for all OS kernels. Please check the release notes.
- ib_ipoib.ko

A.9.1.1 Example: Adding an IB Driver to initrd (Linux)

Prerequisites

1. The FlexBoot image is already programmed on the HCA card.
2. The DHCP server is installed and configured as described in Section 4.6.3.1, "IPoIB Configuration Based on DHCP", and is connected to the client machine.
3. An `initrd` file.
4. To add an IB driver into `initrd`, you need to copy the IB modules to the diskless image. Your machine needs to be pre-installed with a *Mellanox OFED for Linux* ISO image that is appropriate for the kernel version the diskless image will run.

Adding the IB Driver to the initrd File

The following procedure modifies critical files used in the boot procedure. It must be executed by users with expertise in the boot process. Improper application of this procedure may prevent the diskless machine from booting.

Step 1. Back up your current `initrd` file.

Step 2. Make a new working directory and change to it.

```
host1$ mkdir /tmp/initrd_ib
host1$ cd /tmp/initrd_ib
```

Step 3. Normally, the `initrd` image is zipped. Extract it using the following command:

```
host1$ gzip -dc <initrd image> | cpio -id
```

The initrd files should now be found under /tmp/initrd_ib

Step 4. Create a directory for the InfiniBand modules and copy them.

```
host1$ mkdir -p /tmp/initrd_ib/lib/modules/ib
host1$ cd /lib/modules/`uname -r`/updates/kernel/drivers
host1$ cp infiniband/core/ib_addr.ko /tmp/initrd_ib/lib/modules/ib
host1$ cp infiniband/core/ib_core.ko /tmp/initrd_ib/lib/modules/ib
host1$ cp infiniband/core/ib_mad.ko /tmp/initrd_ib/lib/modules/ib
host1$ cp infiniband/core/ib_sa.ko /tmp/initrd_ib/lib/modules/ib
host1$ cp infiniband/core/ib_cm.ko /tmp/initrd_ib/lib/modules/ib
host1$ cp infiniband/core/ib_uverbs.ko /tmp/initrd_ib/lib/modules/ib
host1$ cp infiniband/core/ib_ucm.ko /tmp/initrd_ib/lib/modules/ib
host1$ cp infiniband/core/ib_umad.ko /tmp/initrd_ib/lib/modules/ib
host1$ cp infiniband/core/iw_cm.ko /tmp/initrd_ib/lib/modules/ib
host1$ cp infiniband/core/rdma_cm.ko /tmp/initrd_ib/lib/modules/ib
host1$ cp infiniband/core/rdma_ucm.ko /tmp/initrd_ib/lib/modules/ib
host1$ cp net/mlx4/mlx4_core.ko /tmp/initrd_ib/lib/modules/ib
host1$ cp infiniband/hw/mlx4/mlx4_ib.ko /tmp/initrd_ib/lib/modules/ib
host1$ cp infiniband/hw/mthca/ib_mthca.ko /tmp/initrd_ib/lib/modules/ib
host1$ cp infiniband/ulp/ipoib/ipoib_helper.ko /tmp/initrd_ib/lib/modules/ib
host1$ cp infiniband/ulp/ipoib/ib_ipoib.ko /tmp/initrd_ib/lib/modules/ib
```

Step 5. IB requires loading an IPv6 module. If you do not have it in your `initrd`, please add it using the following command:

```
host1$ cp /lib/modules/`uname -r`/kernel/net/ipv6/ipv6.ko \
/tmp/initrd_ib/lib/modules
```

Step 6. To load the modules, you need the `insmod` executable. If you do not have it in your `initrd`, please add it using the following command:

```
host1$ cp /sbin/insmod /tmp/initrd_ib/sbin/
```

Step 7. If you plan to give your IB device a static IP address, then copy `ifconfig`. Otherwise, skip this step.

```
host1$ cp /sbin/ifconfig /tmp/initrd_ib/sbin
```

Step 8. If you plan to obtain an IP address for the IB device through DHCP, then you need to copy the DHCP client which was compiled specifically to support IB;
Otherwise, skip this step.

To continue with this step, DHCP client v3.1.3 needs to be already installed on the machine you are working with.

Copy the DHCP client v3.1.3 file and all the relevant files as described below.

```
host1# cp <path to DHCP client v3.1.3>/dhclient /tmp/initrd_ib/sbin
host1# cp <path to DHCP client v3.1.3>/dhclient-script /tmp/initrd_ib/sbin
host1# mkdir -p /tmp/initrd_ib/var/state/dhcp
host1# touch /tmp/initrd_ib/var/state/dhcp/dhclient.leases
host1# cp /bin/uname /tmp/initrd_ib/bin
host1# cp /usr/bin/expr /tmp/initrd_ib/bin
host1# cp /sbin/ifconfig /tmp/initrd_ib/bin
host1# cp /bin/hostname /tmp/initrd_ib/bin
```

Create a configuration file for the DHCP client (as described in Section 4.6.3.1) and place it under `/tmp/initrd_ib/sbin`. The following is an example of such a file (called `dclient.conf`):

dhclient.conf:

```
# The value indicates a hexadecimal number
# For a ConnectX device interface "ib0" {send dhcp-client-identifier
ff:00:00:00:00:00:02:00:00:02:c9:00:00:02:c9:03:00:00:10:39;
}
```

Step 9. Now you can add the commands for loading the copied modules into the file `init`. Edit the file `/tmp/initrd_ib/init` and add the following lines at the point you wish the IB driver to be loaded.

The order of the following commands (for loading modules) is critical.

```
echo "loading ipv6"
/sbin/insmod /lib/modules/ipv6.ko
echo "loading IB driver"
```

```
/sbin/insmod /lib/modules/ib/ib_addr.ko
/sbin/insmod /lib/modules/ib/ib_core.ko
/sbin/insmod /lib/modules/ib/ib_mad.ko
/sbin/insmod /lib/modules/ib/ib_sa.ko
/sbin/insmod /lib/modules/ib/ib_cm.ko
/sbin/insmod /lib/modules/ib/ib_uverbs.ko
/sbin/insmod /lib/modules/ib/ib_ucm.ko
/sbin/insmod /lib/modules/ib/ib_umad.ko
/sbin/insmod /lib/modules/ib/iw_cm.ko
/sbin/insmod /lib/modules/ib/rdma_cm.ko
/sbin/insmod /lib/modules/ib/rdma_ucm.ko
/sbin/insmod /lib/modules/ib/mlx4_core.ko
/sbin/insmod /lib/modules/ib/mlx4_ib.ko
/sbin/insmod /lib/modules/ib/ib_mthca.ko
```

The following command (loading ipoib_helper.ko) is *not* required for all OS kernels. Please check the release notes.

```
/sbin/insmod /lib/modules/ib/ipoib_helper.ko
/sbin/insmod /lib/modules/ib/ib_ipoib.ko
```

In case of interoperability issues between iSCSI and Large Receive Offload (LRO), change the last command above as follows to disable LRO:

```
/sbin/insmod /lib/modules/ib/ib_ipoib.ko lro=0
```

Step 10. Now you can assign an IP address to your IB device by adding a call to `ifconfig` or to the DHCP client in the `init` file after loading the modules. If you wish to use the DHCP client, then you need to add a call to the DHCP client in the `init` file after loading the IB modules. For example:

```
/sbin/dhclient -cf /sbin/dhclient.conf ib1
```

Step 11. Save the `init` file.

Step 12. Close `initrd`.

```
host1$ cd /tmp/initrd_ib
host1$ find ./ | cpio -H newc -o > /tmp/new_initrd_ib.img
host1$ gzip /tmp/new_init_ib.img
```

Step 13. At this stage, the modified `initrd` (including the IB driver) is ready and located at `/tmp/new_init_ib.img.gz`. Copy it to the original `initrd` location and rename it properly.

A.9.2 Case II: Ethernet Ports

The Ethernet driver requires loading the following modules in the specified order – see the example below:

- mlx4_core.ko
- mlx4_en.ko

A.9.2.1 Example: Adding an Ethernet Driver to initrd (Linux)

Prerequisites

1. The FlexBoot image is already programmed on the adapter card.
2. The DHCP server is installed and configured as described in Section 4.6.3.1 on page 79, and connected to the client machine.
3. An `initrd` file.
4. To add an Ethernet driver into `initrd`, you need to copy the Ethernet modules to the diskless image. Your machine needs to be pre-installed with a *MLNX_EN Linux Driver* that is appropriate for the kernel version the diskless image will run.

Adding the Ethernet Driver to the initrd File

The following procedure modifies critical files used in the boot procedure. It must be executed by users with expertise in the boot process. Improper application of this procedure may prevent the diskless machine from booting.

Step 1. Back up your current `initrd` file.

Step 2. Make a new working directory and change to it.

```
host1$ mkdir /tmp/initrd_en
host1$ cd /tmp/initrd_en
```

Step 3. Normally, the `initrd` image is zipped. Extract it using the following command:

```
host1$ gzip -dc <initrd image> | cpio -id
```

The initrd files should now be found under /tmp/initrd_en

Step 4. Create a directory for the ConnectX EN modules and copy them.

```
host1$ mkdir -p /tmp/initrd_en/lib/modules/mlnx_en
host1$ cd /lib/modules/`uname -r`/updates/kernel/drivers
host1$ cp net/mlx4/mlx4_core.ko /tmp/initrd_en/lib/modules/mlnx_en
host1$ cp net/mlx4/mlx4_en.ko /tmp/initrd_en/lib/modules/mlnx_en
```

Step 5. To load the modules, you need the `insmod` executable. If you do not have it in your `initrd`, please add it using the following command:

```
host1$ cp /sbin/insmod /tmp/initrd_en/sbin/
```

Step 6. If you plan to give your Ethernet device a static IP address, then copy `ifconfig`. Otherwise, skip this step.

```
host1$ cp /sbin/ifconfig /tmp/initrd_en/sbin
```

Step 7. Now you can add the commands for loading the copied modules into the file `init`. Edit the file `/tmp/initrd_en/init` and add the following lines at the point you wish the Ethernet driver to be loaded.

The order of the following commands (for loading modules) is critical.

```
echo "loading Mellanox ConnectX EN driver"
/sbin/insmod lib/modules/mlnx_en/mlx4_core.ko
/sbin/insmod lib/modules/mlnx_en/mlx4_en.ko
```

Step 8. Now you can assign a static or dynamic IP address to your Mellanox ConnectX EN network interface.

Step 9. Save the `init` file.

Step 10. Close `initrd`.

```
host1$ cd /tmp/initrd_en
host1$ find ./ | cpio -H newc -o > /tmp/new_initrd_en.img
host1$ gzip /tmp/new_init_en.img
```

At this stage, the modified `initrd` (including the Ethernet driver) is ready and located at `/tmp/new_init_ib.img.gz`. Copy it to the original `initrd` location and rename it properly.

A.10 iSCSI Boot

Mellanox FlexBoot enables an iSCSI-boot of an OS located on a remote iSCSI Target. It has a built-in iSCSI Initiator which can connect to the remote iSCSI Target and load from it the kernel and `initrd` (Linux). There are two instances of connection to the remote iSCSI Target: the first is for getting the kernel and `initrd` via FlexBoot, and the second is for loading other parts of the OS via `initrd`.

If you choose to continue loading the OS (after boot) through the HCA device driver, please verify that the initrd image includes the HCA driver as described in Section A.8.

A.10.1 Configuring an iSCSI Target in Linux Environment

Prerequisites

Step 1. Make sure that an iSCSI Target is installed on your server side.

You can download and install an iSCSI Target from the following location:
http://sourceforge.net/projects/iscsitarget/files/iscsitarget/

Step 2. Dedicate a partition on your iSCSI Target on which you will later install the operating system

Step 3. Configure your iSCSI Target to work with the partition you dedicated. If, for example, you choose partition /dev/sda5, then edit the iSCSI Target configuration file `/etc/ietd.conf` to include the following line under the iSCSI Target iqn line:

```
Lun 0 Path=/dev/sda5,Type=fileio
```

Example of an iSCSI Target iqn line:

```
Target iqn.2007-08.7.3.4.10:iscsiboot
```

Step 4. Start your iSCSI Target.

Example:

```
host1# /etc/init.d/iscsitarget start
```

Configuring the DHCP Server to Boot From an iSCSI Target

Configure DHCP as described in Section 4.6.3.1, "IPoIB Configuration Based on DHCP".

Edit your DHCP configuration file (`/etc/dhcpd.conf`) and add the following lines for the machine(s) you wish to boot from the iSCSI target:

```
Filename "";
option root-path "iscsi:iscsi_target_ip::::iscsi_target_iqn";
```

The following is an example for configuring an IB/ETH device to boot from an iSCSI target:

```
host  host1{
filename "";

# For a ConnectX device with ports configured as InfiniBand, comment out# the following
line
# option dhcp-client-identifier =
ff:00:00:00:00:00:02:00:00:02:c9:00:00:02:c9:03:00:00:10:39;

# For a ConnectX device with ports configured as Ethernet, comment out# the following
line
# hardware ethernet 00:02:c9:00:00:bb;

}
```

A.11 WinPE

Mellanox FlexBoot enables WinPE boot via TFTP. For instructions on preparing a WinPE image, please see http://etherboot.org/wiki/winpe.

Appendix B: SRP Target Driver

The SRP Target driver is designed to work directly on top of OpenFabrics OFED software stacks (http://www.openfabrics.org) or InfiniBand drivers in Linux kernel tree (kernel.org). It also interfaces with Generic SCSI target mid-level driver - SCST (http://scst.sourceforge.net).

By interfacing with an SCST driver, it is possiblee to work with and support a lot of IO modes on real or virtual devices in the backend.

1. scst_vdisk – fileio and blockio modes. This allows turning software raid volumes, LVM volumes, IDE disks, block devices and normal files into SRP luns
2. NULLIO mode allows measuring the performance without sending IOs to *real* devices

B.1 Prerequisites and Installation

1. SRP targer is part of the OpenFabrics OFED software stacks. Use the latest OFED distribution package to install SRP target.

On distribution default kernels you can run scst_vdisk blockio mode to obtain good performance.

2. Download and install the SCST driver. The supported version is 1.0.1.1.
   a. Download `scst-1.0.1.1.tar.gz` from http://scst.sourceforge.net/downloads.html
   b. Untar `scst-1.0.1.1`

```
$ tar zxvf scst-1.0.1.1.tar.gz
$ cd scst-1.0.1.1
```

   c. Install `scst-1.0.1.1 as follows:`

```
$ make && make install
```

B.2 How-to run

A. On an SRP Target machine:

1. Please refer to SCST's README for loading scst driver and its dev_handlers drivers (scst_vdisk block or file IO mode, nullio, ...)

Regardless of the mode, you always need to have lun 0 in any group's device list. Then you can have any lun number following lun 0 (it is not required to have the lun numbers in ascending order except that the first lun must always be 0).

Setting SRPT_LOAD=yes in /etc/infiniband/openib.conf is not enough as it only loads the `ib_srpt` module but does *not* load `scst` not its dev_handlers.

The scst_disk module (pass-thru mode) of SCST is not supported by Mellanox OFED.

Example 1: Working with VDISK BLOCKIO mode

(Using the md0 device, sda, and cciss/c1d0)

a. modprobe scst
b. modprobe scst_vdisk
c. echo "open vdisk0 /dev/md0 BLOCKIO" > /proc/scsi_tgt/vdisk/vdisk
d. echo "open vdisk1 /dev/sda BLOCKIO" > /proc/scsi_tgt/vdisk/vdisk
e. echo "open vdisk2 /dev/cciss/c1d0 BLOCKIO" > /proc/scsi_tgt/vdisk/vdisk
f. echo "add vdisk0 0" >/proc/scsi_tgt/groups/Default/devices
g. echo "add vdisk1 1" >/proc/scsi_tgt/groups/Default/devices
h. echo "add vdisk2 2" >/proc/scsi_tgt/groups/Default/devices

Example 2: working with scst_vdisk FILEIO mode

(Using md0 device and file 10G-file)

a. modprobe scst
b. modprobe scst_vdisk
c. echo "open vdisk0 /dev/md0" > /proc/scsi_tgt/vdisk/vdisk
d. echo "open vdisk1 /10G-file" > /proc/scsi_tgt/vdisk/vdisk
e. echo "add vdisk0 0" >/proc/scsi_tgt/groups/Default/devices
f. echo "add vdisk1 1" >/proc/scsi_tgt/groups/Default/devices

2. Run:

```
For all distributions except SLES 11:  > modprobe ib_srpt
For SLES 11:   > modprobe -f ib_srpt
```

For SLES 11, please ignore the following error messages in /var/log/messages when loading ib_srpt to SLES 11 distribution's kernel:

```
...
ib_srpt: no symbol version for scst_unregister
ib_srpt: Unknown symbol scst_unregister
```

```
ib_srpt: no symbol version for scst_register
ib_srpt: Unknown symbol scst_register
ib_srpt: no symbol version for scst_unregister_target_template
ib_srpt: Unknown symbol scst_unregister_target_template
...
```

B. On Initiator Machines

On Initiator machines, manually perform the following steps:

1. Run:

```
modprobe ib_srp
```

2. Run:

```
ibsrpdm -c -d /dev/infiniband/umadX
```

(to discover a new SRP target)

```
umad0: port 1 of the first HCA
umad1: port 2 of the first HCA
umad2: port 1 of the second HCA
```

3. echo {new target info} > /sys/class/infiniband_srp/srp-mthca0-1/add_target
4. fdisk -l (will show the newly discovered scsi disks)

Example:

Assume that you use port 1 of first HCA in the system, i.e.: mthca0

```
[root@lab104 ~]# ibsrpdm -c -d /dev/infiniband/umad0
id_ext=0002c90200226cf4,ioc_guid=0002c90200226cf4,
dgid=fe8000000000000002c90200226cf5,pkey=ffff,service_id=0002c90200226cf4
[root@lab104 ~]# echo id_ext=0002c90200226cf4,ioc_guid=0002c90200226cf4,
dgid=fe8000000000000002c90200226cf5,pkey=ffff,service_id=0002c90200226cf4 > /sys/
class/infiniband_srp/srp-mthca0-1/add_target
```

OR

- You can edit /etc/infiniband/openib.conf to load the SRP driver and SRP High Availability (HA) daemon automatically, that is: set “SRP_LOAD=yes” and “SRPHA_ENABLE=yes”
- To set up and use the HA feature, you need the dm-multipath driver and multipath tool
- Please refer to OFED-1.x SRP's user manual for more detailed instructions on how-to enable/use the HA feature

The following is an example of an SRP Target setup file:

```
********************** srpt.sh *******************************
#!/bin/sh
modprobe scst scst_threads=1
modprobe scst_vdisk scst_vdisk_ID=100
```

```
echo "open vdisk0 /dev/cciss/c1d0 BLOCKIO" > /proc/scsi_tgt/vdisk/vdisk
echo "open vdisk1 /dev/sdb BLOCKIO" > /proc/scsi_tgt/vdisk/vdisk
echo "open vdisk2 /dev/sdc BLOCKIO" > /proc/scsi_tgt/vdisk/vdisk
echo "open vdisk3 /dev/sdd BLOCKIO" > /proc/scsi_tgt/vdisk/vdisk
echo "add vdisk0 0" > /proc/scsi_tgt/groups/Default/devices
echo "add vdisk1 1" > /proc/scsi_tgt/groups/Default/devices
echo "add vdisk2 2" > /proc/scsi_tgt/groups/Default/devices
echo "add vdisk3 3" > /proc/scsi_tgt/groups/Default/devices

modprobe ib_srpt

echo "add "mgmt"" > /proc/scsi_tgt/trace_level
echo "add "mgmt_dbg"" > /proc/scsi_tgt/trace_level
echo "add "out_of_mem"" > /proc/scsi_tgt/trace_level
********************** End srpt.sh ****************************
```

B.3 How-to Unload/Shutdown

1. Unload `ib_srpt`

```
$ modprobe -r ib_srpt
```

2. Unload `scst` and its dev_handlers first

```
$ modprobe -r scst_vdisk scst
```

3. Unload `ofed`

```
$ /etc/rc.d/openibd stop
```

Appendix C: mlx4 Module Parameters

In order to set **mlx4** parameters, add the following line(s) to **/etc/modprobe.conf**:

```
options mlx4_core parameter=<value>
```

and/or

```
options mlx4_ib   parameter=<value>
```

and/or

```
options mlx4_en   parameter=<value>
```

and/or

```
options mlx4_fc   parameter=<value>
```

The following sections list the available `mlx4` parameters.

C.1 mlx4_core Parameters

```
set_4k_mtu:          Attempt to set 4K MTU to all ConnectX ports (int)
pfctx:              Priority based Flow Control policy on TX[7:0]. Per priority bit mask
                     (uint)
pfcrx:             Priority based Flow Control policy on RX[7:0]. Per priority bit mask
                     (uint)
debug_level:         Enable debug tracing if > 0 (int)
block_loopback:      Block multicast loopback packets if > 0 (int)
msi_x:               Attempt to use MSI-X if nonzero (int)
high_rate_steer:     Use mac table steering for Ethernet ports (default 0) (int)
log_num_mac:         Log2 max number of MACs per ETH port (1-7) (int)
use_prio:            Enable steering by VLAN priority on ETH ports (0/1, default 0)
(bool)
fast_drop:           Enable fast packet drop when no recieve WQEs are posted (int)
log_num_qp:          Log maximum number of QPs per HCA (int)
log_num_srq:         Log maximum number of SRQs per HCA (int)
log_rdmarc_per_qp:   Log number of RDMARC buffers per QP (int)
log_num_cq:          Log maximum number of CQs per HCA (int)
log_num_mcg:         Log maximum number of multicast groups per HCA (int)
log_num_mpt:         Log maximum number of memory protection table entries per HCA (int)
log_num_mtt:         Log maximum number of memory translation table segments per HCA
(int)
log_mtts_per_seg:    Log2 number of MTT entries per segment (0-7) (int)
enable_qos:          Enable Quality of Service support in the HCA (default: off) (bool)
enable_pre_t11_mode:For FCoXX, enable pre-t11 mode if non-zero (default: 0) (int)
internal_err_reset: Reset device on internal errors if non-zero (default 1) (int)
```

C.2 mlx4_ib Parameters

```
debug_level                      Enable debug tracing if > 0 (default 0)
```

C.3 mlx4_en Parameters

```
 inline_thold                 Threshold for using inline data (int)
 num_rx_rings                 Total number of RX Rings (default 16, range 1-16, power
                                of 2) (uint)
 udp_rss                        Enable RSS for incoming UDP traffic or disabled (0)
(bool)
 num_lro                      Number of LRO sessions per ring or disabled (0) (unit)
 use_napi                     Use NAPI (1 - default) or process incoming traffic from
                                interrupt context (0) (bool)
 use_tx_polling                Use polling for TX processing (default 1) (bool)
 enable_sys_tune              Tune the cpu's for better performance (default 0) (bool)
```

Appendix D: ib-bonding Driver for Systems Using SLES10 SP4

D.1 Using the ib-bonding Driver

The ib-bonding driver is a High Availability solution for IPoIB interfaces. It is based on the Linux Ethernet Bonding Driver and was adapted to work with IPoIB. The ib-bonding package contains a bonding driver and a utility called ib-bond to manage and control the driver operation.

The ib-bonding driver comes with the ib-bonding package (run "`rpm -qi ib-bonding`" to get the package information).

The ib-bonding driver can be loaded manually or automatically.

Manual Operation

Use the utility `ib-bond` to start, query, or stop the driver. For details on this utility, please read the documentation for the ib-bonding package under

```
/usr/share/doc/ib-bonding-0.9.0/ib-bonding.txt on RedHat, and
/usr/share/doc/packages/ib-bonding-0.9.0/ib-bonding.txt on SuSE.
```

Automatic Operation

Automatic ib-bonding operation can be configured as follow:

1. Using a standard OS bonding configuration. For details on this, please read the documentation for the ib-bonding package under

```
/usr/share/doc/ib-bonding-0.9.0/ib-bonding.txt on RedHat, and
/usr/share/doc/packages/ib-bonding-0.9.0/ib-bonding.txt on SuSE.
```

Notes

- If the bondX name is defined but one of bondX_SLAVES or bondX_IPs is missing, then that specific bond will not be created.
- The bondX name must not contain characters which are disallowed for bash variable names such as '.' and '-'.

All the newer OSes Bonding can be done with the inbox bonding module.